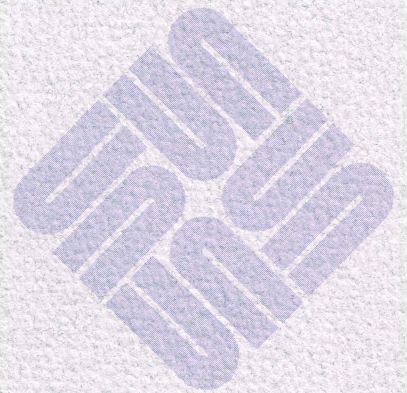




Software Technical Bulletin

November 1989

Technical Information Services



Part Number 812-8911-01
Issue 1989-11
November 1989



Software Technical Bulletin

November 1989

Technical Information Services

Software Technical Bulletins are distributed to customers with software/hardware or software only support contracts. Send comments or corrections to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Ave., M/S 2-318, Mountain View, CA 94043 or by electronic mail to *sun/stb-editor*. U.S. customers who have technical questions about topics in the Bulletin should call the Sun Customer Software Services AnswerLine at 800 USA-4-SUN. Other customers should call the numbers listed in *World Hotlines* appearing in Section 1.

Sun-2, Sun-2/xxx, Sun-3, Sun-4, Sun386i, Deskside, SunStation, Sun Workstation, SunCore, SunPHIGS, DVMA, SunWindows, NeWS, NFS, NSE, SPARC™, SunUNIFY™, SunView™, SunGKS, SunCGI, SunGuide, SunSimplify, SunLink, Sun Microsystems, SunOS™, TOPS®, Flashcaard™, SunPaint™, SunWrite™, SunDraw™, TOPS Terminal, View2, and the Sun logo are trademarks of Sun Microsystems, Inc. UNIX, UNIX/32V, UNIX System III, UNIX System V, and OPEN LOOK are trademarks of AT&T Bell Laboratories. DEC, DNA, VAX, VMS, VT100, WPS-PLUS, MicroVAX, and Ultrix are registered trademarks of Digital Equipment Corporation. Courier 2400 is a trademark of U.S. Robotics, Inc. Hayes is a trademark of Hayes Microcomputer Products, Inc. Multibus is a trademark of Intel Corporation. PostScript and TranScript are trademarks of Adobe Systems, Inc. Ven-Tel is a trademark of Ven-Tel, Inc. UNIFY™ is a trademark of Unify Corporation. ENTER, PAINT, ACCELL, and RPT are trademarks of Unify Corporation. IBM, IBM PC, PC, IBM PC/XT, IBM PC/AT, and SQL™ are registered trademarks of International Business Machines Corporation. Applix® is a registered trademark of Applix, Inc. SunAlis™ is a trademark of Sun Microsystems, Inc. and is derived from Alis, a product marketed by Applix, Inc. SunINGRES™ is a trademark of Sun Microsystems, Inc. and is derived from INGRES, a product marketed by Relational Technology, Inc. VEGA Delux is a trademark of Video Seven, Inc. Micro Enhancer Delux is a trademark of Everex Systems, Inc. VxWorks is a trademark of Wind River Systems, Inc. Cabletron is a trademark of Cabletron Systems. Apple, Finder, Macintosh, Appletalk, MacWrite, and Laser Writer™ are registered trademarks of Apple Computer, Inc. PostScript® is a registered trademark of Adobe Systems, Inc. Excel, MS-DOS, and Microsoft Word are registered trademarks of Microsoft Corporation. Fastpath is a trademark of Kinetics, Inc. Ethernet is a registered trademark of Xerox Corporation. Lotus 1-2-3 is a trademark of Lotus Development Corporation. Word Perfect is a trademark of the Word Perfect Corporation. Wyse-50 is a trademark of Wyse Technology Corporation. AMD 7990 LAN Controller is a trademark of Advanced Micro Devices, Incorporated. Proximity (R) is a registered trademark of Proximity Technology, Inc. Merriam-Webster (R) is a registered trademark of Merriam-Webster Inc. SPAM (R) is a registered trademark of a pork product packed only by Geo A. Hormel & Co. Corp. YELLOW PAGES (R) is a registered trademark of British Telecommunications Plc.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Copyright (c) 1989, Sun Microsystems, Inc. Printed in U.S.A. All Rights Reserved. No part of this work covered by copyright hereon may be reproduced or used in any form or by any means -- graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems -- without permission of the copyright owner.

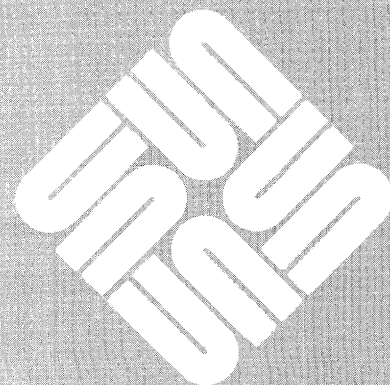
RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

Contents

Section 1 NOTES & COMMENTS	1395
Editor's Notes	1395
World Hotlines	1397
Reporting Bugs	1399
STB Duplication	1407
Section 2 ARTICLES	1411
calendar Network Traffic	1411
Diskless-Client Booting	1413
Section 3 STB SHORT SUBJECTS	1419
Selection Service Messages	1419
valloc failed Defined	1420
Section 4 IN DEPTH	1423
Sun386i Administration Cookbook: Ch. 9, Under the Hood	1423
Section 5 HINTS AND TIPS	1427
Sun386i Backups	1427
DOS Memory Tips	1429
Section 6 THE HACKERS' CORNER	1433
FreeSpace	1433
Section 7 HARDWARE, CONFIGURATIONS, & UPGRADES	1441
Sun386i Ethernet Pins	1441
Sun386i Serial Pins	1442
Software Release Levels	1443
Section 8 CUMULATIVE INDEX: 1989	1451

NOTES & COMMENTS

NOTES & COMMENTS	1395
Editor's Notes	1395
World Hotlines	1397
Reporting Bugs	1399
STB Duplication	1407



NOTES & COMMENTS

Editor's Notes

Editor's Notes

The editor's notes for this November 1989 issue include the items of interest listed below.

- In Depth Special Features: The Sun386i Administration Cookbook
- World hotlines for customer service calls
- World-wide bug reporting information
- Limited permission to duplicate your STB
- Hints and Tips: Saving backup time and determining DOS memory
- The Hackers' Corner: FreeSpace
- Configurations: updated software release level tables, effective September 22, 1989

In Depth Special Features: The Sun386i Administration Cookbook

This month begins a three-month series of In Depth features which will include the *Sun386i Administration Cookbook*. This first month includes the longest chapter of immediate use to those customers running Sun386i workstations: Chapter 9, Under the Hood.

Future STB issues will include the cookbook table of contents, preface, the remaining chapters, an automounter appendix, and a separate cookbook index.

STB readers should note that the pagination in the upcoming In Depth features is the same as in the cookbook. Simply remove the cookbook pages from the November, December, and January 1990 STBs and insert them in a separate Sun386i Administration Cookbook binder. The remaining STB pages are paginated cumulatively as reflected in the STB Cumulative Index.

World Hotlines

For Sun customers world-wide served by your local service groups, use the customer service telephone numbers listed in this monthly item. Also, look to this section during the upcoming year for details on your local support call policies and procedures.

Reporting Bugs World-Wide

A list of Sun service centers, addresses, email hotlines, and telephone hotlines appears. The information in this monthly note continues to be expanded as Sun software service centers are added world-wide.

STB Duplication Permission

This notice is published monthly, giving customers useful information regarding ordering and duplicating additional STB copies. This duplication permission is limited, as detailed in the note.

Hints and Tips

This month's hints and tips section contains two new items of interest. The first is a hint on how to reduce the time needed to do backups for those running Sun386i SunOS 4.0.2 on Sun386i machines.

This month's tips are on how to determine available DOS memory on Sun386i machines, and possible uses of DOS extenders.

The Hackers' Corner

This month's **Hackers' Corner** contains a FreeSpace tool of use to those wishing to sort through files to find which ones may be deleted to free disk space.

For those with email access and wishing an online copy of **Hackers' Corner** code samples, please email *sun!stb-editor* or *stb-editor@sun* with your request. Please include the program title, and the STB issue month and year with your request.

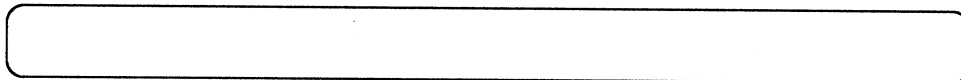
Again, please note that such applications, scripts, or code are not offered as released Sun products, but as items of interest to enthusiasts wanting to try out something for themselves. They may not work in all cases, and may not be compatible with future SunOS releases. Please consult your local shell script or programming expert regarding any application, script, or code problems.

Configurations: Current Sun Software Products and Release Level Tables

The seven tables showing current Sun software product release levels appear monthly. These tables show release levels for operating systems, communications products, unbundled languages, unbundled applications, unbundled graphics, other products, and TOPS networking products. The tables in this issue are updated through September 22, 1989.

Thanks.

The STB Editor

World Hotlines**World Hotlines**

Sun Customers throughout the world have service hotlines available for both software and hardware support questions. The service hotlines are shown below. If your country is not shown in the table, please phone your local Sun sales office.

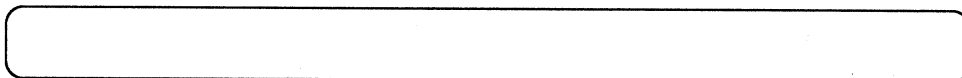
The world hotlines are divided into those for Canada and the USA, CSD Europe, and Intercon. Intercon includes those countries outside the USA, Canada, Europe, and northern Africa.

Canada and the United States

Canada	Montreal	(514) 738-4885
	Ottawa	(613) 723-8112
	Toronto	(416) 477-6745
	Winnipeg	(204) 222-2333
	Edmonton	(403) 482-7264
	Calgary	(403) 262-6722
	Vancouver	(604) 684-4120
United States	All, including Puerto Rico	1-800-USA-4-SUN
CSD Europe		
European Customer Service	Surrey Sun Microsystems Europe Inc.	(44) 276 51440
France	Paris Sun Microsystems France SA	(33) 1 4094 8080
Germany	Munich Sun Microsystems GmbH	(49) 089/46008-321
The Netherlands	Soest Sun Microsystems Nederland BV	(31) 2155 24888
Sweden	Solna Sun Microsystems AB	+46 8 764 78 10
Switzerland	Zurich Sun Microsystems (Schweiz) AG	(41) 1 828 9555
United Kingdom	Albany Park Sun Microsystems UK Ltd	(44) 0276 691052

Intercon Australia	Sun Microsystems Australia	(011-61-2) 436-4699
Hong Kong	Sun Hong Kong	(011-852-5) 865-1688
Japan	C. Itoh Data Systems Nihon Sun	(011-81-3) 497-4676 (011-81-3) 221-7021
Countries Not Listed	All countries outside the USA, Canada, Europe, northern Africa, Australia, and Japan	(415) 496-6119

Reporting Bugs



Submitting Bugs and Email Service Calls

This article contains two sections for submitting bugs. The first section describes procedures to use within the United States. The second section describes Customer Service Division (CSD) Europe procedures.

Submitting Software Bugs: United States and Canada

This section contains information on reporting bugs within the U.S., for customers holding and not holding support contracts.

Sun's United States Answer Center (USAC) within CSD accepts software bug reports from Sun users via electronic mail and by phone. The method you use to submit a bug report varies with your needs.

U.S. users holding support contracts can report bugs to USAC via the (800) USA-4-SUN phone hotline. Canadian users holding support contracts should call their local support center. The USAC phone hotline is the fastest way for a customer to find out if a problem is known and if a workaround exists. The status of previously-reported bugs can also be obtained in this way. The list of open software bugs is contained in the Customer Distributed BugsList (CDB).

Customers holding support contracts can also submit bug reports electronically to the address *sun!hotline* (*hotline@sun.COM*). This method generates a service order, and can be used when lines of code or other information difficult to relay over the phone is needed to describe the bug.

- Whenever possible, customers should use the Online Bugs Database (OBD) described below before submitting bugs, to avoid resubmitting an already-known bug.
- Please note, however, that the alias *onlinebugs-db@sun.com* is **not** the appropriate avenue for submitting bugs.

Customers who do not hold Sun software support contracts can report bugs via electronic mail to the address *sun!sunbugs* (or *sunbugs@sun.COM*). These reports are reviewed periodically to determine proper disposition. Those reports determined to be from supported customers are forwarded to the U.S. Answer Center for handling. Reports from customers who cannot be verified as holding a support contract are reviewed by Sun's engineering and support personnel. An internal bug report is generated if the reported bug is new and verifiable.

Finally, customers not holding software support contracts may call the (800) USA-4-SUN phone hotline to report a problem and request support on a Time and Materials (T&M) basis. Canadian customers should call their local support center. In this case, please have a Purchase Order (PO) number for billing purposes.

The Online Bugs Database (OBD)

The OBD contains the same information as the Customer Distributed BugsList (CDB). The information available through the OBD is updated during the first week of each month. As a result, you receive the most timely information available on open known bugs and temporary workarounds for Sun software in an easily-accessible, online format.

The OBD service is initially available only within the United States. Future plans include worldwide introduction and distribution.

Information Provided by the OBD

The OBD provides you with rapid telephone access to the following information.

- Software Bug Reference Number--a unique identification number assigned to each valid software bug by Sun
- Online Bug Synopsis--a one-line summary of the software bug
- Bug Description--a brief description of the bug, with examples if available
- Software release(s) in which the bug was reported
- Affected configurations
- Temporary workarounds, where available

To use the OBD, simply dial the telephone number and enter the system password; both provided in the *Online Bugs Database Reference Manual*, part number 812-1001. This manual is automatically sent to the site contact of all Sun customers holding valid support contracts. The OBD is available at all hours, except for scheduled updates and preventive maintenance. System support is available during standard U.S. Answer Center business hours by calling the support numbers given above.

OBD Search Criteria

After logging in, you can quickly search the OBD by any one of the below parameters.

- Keyword(s)
- Software Bug Reference Number
- Software Category (such as kernel, SunINGRES, or Datacomm)
- Software Subcategory (such as documentation related to a specific category)
- Software Release (such as 4.0, 3.5, 3.4, 3.2, 3.0)

Search capabilities can be enhanced by combining several of the primary search parameters. For example, all release 3.4 NFS bugs within the network category can be searched. In most situations, you can locate a particular software bug and its related workaround within 30 seconds.

To ensure that your OBD use is as efficient as possible, a fast, easy-to-use Help facility is also provided. Help is available throughout your OBD session.

**Summary: United States and
Canada**

For U.S. contract customers, (800) USA-4-SUN is the best method to report bugs. Canadian contract customers should report bugs to their local support center. The electronic mail address *sun!hotline* is available to submit materials that are difficult to relay over the phone. The OBD is available to research currently-known bugs.

For non-contract customers, the electronic mail address *sun!sunbugs* is available to report bugs.

To help us serve you better, please include the following information with all electronic mail reports.

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing
- Please do *not* submit bugs to *sun!onlinebugs-db*

**Submitting Software Bugs:
CSD Europe**

This section contains information on reporting bugs within CSD Europe, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

Summary: CSD Europe

To help CSD Europe service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Detailed information for European Customer Service and individual countries follows.

European Customer Service

The European Customer Service office is located at the address shown below.

Sun Microsystems Europe, Inc.
Bagshot Manor
Green Lane
BAGSHOT
Surrey GU19 5NL
United Kingdom

Telephone: (44) 276 51440

Telefax: (44) 276 51287

Telex: 859017

France

Report bugs to the France Answer Center at the postal address shown below.

Service "HOT LINE"
SUN Microsystems France
La Boursidiere
R.N. 186
92357 Le Plessis Robinson Cedex

Hotline Telephone: (33) 1 4094 8080

Telefax: 0276 691774

Special Dispatch Arrangements:

Please provide Dispatch with the following items:

System serial number *or* Contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and

Materials (T&M) basis, and order this support at the above address.

Germany

Report bugs to the Germany Answer Center at the postal address shown below.

Hotline
Sun Microsystems GmbH
Stoerungsannahme
Am Hochacker 3
D-8011 Grasbrunn 1
West-Germany

Hotline Telephone: (49) 089/46008-321

Telex: 5 218 197 sun

Telefax: 089/46008-400

Email Address: *{sunuk,unido}!sunmuc!hotline*

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

The Netherlands

Report bugs to The Netherlands Answer Center at the postal address shown below.

Sun Microsystems Nederland BV
Birkstraat 95-97
3768 HD SOEST
The Netherlands

Hotline Telephone: (31) 2155 24888

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Sweden

Report bugs to the Sweden Answer Center at the postal address shown below.

Sun Microsystems AB
Hemvarnsgatan 9
S 171 54 Solna
Sweden

Hotline Telephone: +46 8 764 78 10

Email Address: *hotline@sunswe.se* or *sunswe!hotline*

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Switzerland

Report bugs to the Switzerland Answer Center at the postal address shown below.

Sun Microsystems (Schweiz) AG
Postfach
Rohrstrasse 36/38
CH-8152 GLATTBRUGG
Switzerland

Hotline Telephone: (41) 1 828 9555

Email Address: *sunuk!sunswis!hotline*

Special Dispatch Arrangements:

Provide Dispatch with the following item:

Contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

United Kingdom

Report bugs to the UK Answer Center at the postal address shown below.

Hotline
Sun Microsystems (UK) Ltd
Technical Centre
Unit 3D
Albany Park
Frimley
Surrey
GU15 2PL

Hotline Telephone: (44) 0276 691052

Telefax: 0276 691774

Special Dispatch Arrangements:

Please provide Dispatch with the following items:

System serial number *or* contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

**Submitting Software Bugs:
Intercon**

This section contains information on reporting bugs within Intercon, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

Summary: Intercon

To help Intercon service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Detailed information for individual countries follows.

Australia

Report bugs to the Australian Answer Center at the postal address shown below.

Hotline
Sun Microsystems Australia Pty Ltd
PO Box 320
Artarmon
NSW 2064

Hotline Telephone: (011-61-2) 436-4699

Telefax: 02 436 1084

Special Dispatch Arrangements:

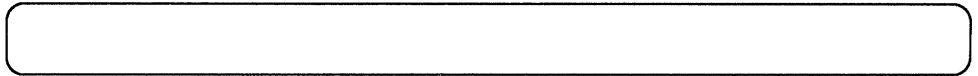
Please provide Dispatch with the following items:

System serial number *or* contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

STB Duplication



Duplicating the STB

Your company's software support contract includes a monthly issue of the STB. Each month, the copy of your STB is mailed to your company's primary contact person or department. Sites with more than one contract may receive more than one STB copy, depending on how the contracts are set up.

Your primary contact person or department may duplicate this 'master' STB copy for all Sun workstation end-users. So long as you duplicate copies and route them only internally, there are no copyright infringement problems.

This limited permission for duplication is for your convenience only, however, and does not include any duplication for resale, for distribution outside your company, or for distribution to employees of companies not having a Sun software support contract.

Direct STB Purchase

The STB is sent to the primary contact person named in all software support contracts. Sun is looking into methods by which customers holding these contracts may purchase extra copies directly.

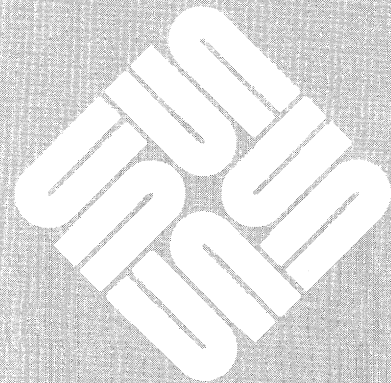
Look to this column for an announcement regarding the purchase of extra STB copies.

Further Questions

If you have any questions, comments, or articles regarding the STB or CDB, please send your ideas and questions to *sun!stb-editor*.

ARTICLES

ARTICLES	1411
calendar Network Traffic	1411
Diskless-Client Booting	1413



ARTICLES

calendar Network Traffic

/bin/calendar Network Traffic

In SunOS 4.x releases, `/bin/calendar` does a `ypcat` as well as `cat /etc/passwd` to determine all possible users who have a `~/calendar` file. There are three problems caused by this process.

Problem 1

Every 4.x machine is programmed in the `crontab` to wake up at midnight and perform `ypcat`. This triggers `/bin/calendar` to be executed for every `ypcat` entry. This affects the network traffic and YP slaves.

Problem 2

When running `automount` to mount all possible home directories, the `/bin/calendar` tries to mount all of them, causing extra network load. When an NFS mount fails, the machine hangs because `/home` directories are typically mounted `rw,hard`.

`calendar` does not function for a diskless client because all filesystems are NFS. Note that `'calendar -'` reports only if the `~/calendar` is in a local filesystem.

Problem 3

Any machine that uses `/etc/passwd` as the master ASCII file reads the same information from `ypcat passwd` and `cat /etc/passwd`. This doubles the amount of `calendar`'s execution, and affects the network traffic and YP slaves.

The Workarounds

The following are offered as suggested workarounds.

Diskless Clients

- Disable the `'calendar -'` line from root's `crontab`.

Diskful Machine

- Disable the `'calendar -'` line from root's `crontab`.
- If the home directory is on a local disk, run `calendar` to create the user's own `crontab` and include `/bin/calendar`.

Servers

For servers, `grep /'hostname'/` from the `ypcat` by replacing the line below:

```
caldata="/bin/ypcat passwd.byname"
```

with:

```
caldata="/bin/ypcat passwd.byname | grep /'hostname'/"
```

and by replacing the following line:

```
$caldata |cat /etc/passwd - | grep -v '^[+-]' |\
```

with:

```
eval $caldata |cat /etc/passwd - | grep -v '^[+-]' |\
```

YP Masters

For YP masters using `/etc/passwd` as the master ASCII file, replace:

```
$caldata |cat /etc/passwd - | grep -v '^[+-]' |\
```

with:

```
grep /'hostname'/ /etc/passwd - | grep -v '^[+-]' |\
```

Diskless-Client Booting

Diskless-Client Booting: Debugging Network-Related Problems

In debugging client boot problems, it is important to determine in what stage the failure exists and then debug from there. The three primary stages of the diskless-client boot are listed below.²

1. The client PROM initiates a reverse `arp (rarp)` request to determine its IP address, and then executes a `tftp` implementation to load the client boot program from a remote server (`/tftpboot/file`).
2. The client boot program implements RPC requests interacting with a `bootparamd` server to determine the network location of its remote disk filesystems (e.g. `root` and `swap`), and then invokes the NFS protocol to mount a remote `root` filesystem and load `vmunix` .
3. `vmunix` loads and initializes.

Generally the console messages on the booting client suggest the current boot stage being processed and where the failure is occurring if the boot hangs. Debugging suggestions for the three boot stages are detailed below.

Stage 1 Debugging

A hang in stage 1 occurs when the diskless client boot cannot get an IP address, or gets an IP address but then fails during the `tftp` download of a boot program. The hang suggests that there are either server-side configuration problems or network problems that prohibit proper client/server interaction on the ethernet.

Network packet tracing should be considered when related server configuration files (i.e. `ethers`, `hosts`) and `tftpboot` links have been verified as correct, and the dependent server processes are available (i.e. `rarpd`, `in.tftpd`, `inetd`). Packet tracing can be done easily by using the `etherfind(8C)` command if another Sun machine that can serve as network monitor is on the same IP network.

A failure with the `rarp` process will generally result in the below message appearing on the client console.

Requesting Internet Address for *ethernet address*

² This article is submitted by Mark Allen, Manager, Data Comm Group, U.S. Answer Center, Mountain View, California, USA.

The above message suggests that the client is not receiving an `rarpd` reply to its request. As root on the Sun monitor machine, invoke the following command to monitor `rarp` interaction between the diskless client and a server. Then boot the the diskless client.

```
# etherfind -i interface -rarp
```

The above command displays `rarp` packet activity. In a successful case, there should be an initial client `rarp` broadcast followed by a server reply to the client. If you see no activity, client ethernet problems are indicated. If a client broadcast is seen without a server reply, other ethernet problems affecting the server's receiving the broadcast or problems affecting the server's reply to the broadcast are indicated.

After a successful `rarp` process, the client will make a `tftp` request to obtain a boot block. Failure of the `tftp` process will generally result in the following message on the client console.

```
tftp: time-out
```

Use the following `etherfind` command and arguments on the Sun monitor machine to monitor the `tftp` activity. The command monitors client-initiated `tftp` activity and any server responses.

```
# etherfind -i interface -proto udp -src clientname -o -dst clientname
```

The same debugging logic applies here as in the previous `rarp` example. You are looking for expected client/server interaction, or otherwise noticing the direction of packet activity to determine the point of failure.

Also look for any unexpected responses that might indicate incorrect server interaction. It is possible to have more than one machine on the net running `rarpd`. It is possible for more than one machine to reply to the client's `rarp` request in cases where common `ethers` and `hosts` file information is shared.

In such cases some booting delays occur since the client will first try to obtain a boot block by sending the `tftp` request to the machine that replied to the client's `rarp` request. However, that machine may not also be the `tftpboot` server for the client. The client then broadcasts again to the network, looking for its `tftpboot` server.

This rebroadcast can cause some client boot process delays. The first example `etherfind` command shown above uses the `-rarp` option and would show this case. It may help in determining if there are some unnecessary `rarpd` processes responding on the network.

Stage 2 Debugging

Debugging failures in stage 2 includes those failures that occur after a client successfully boots through the `rarp` and `tftp` processes (stage 1), but fails prior to actually booting a `vmunix` (stage 3). A failure at this stage usually indicates problems during client interaction with the server's `rpc.bootparamd` process.

There are a number of interactions that occur during this stage. The best way to debug failures at this stage is to do the following.

1. On the server side, kill and restart `rpc.bootparamd` with the debug `-d` option to the `bootparamd(8)` command. This may provide more verbose debugging information to help you determine why the failure is occurring.
2. On the Sun monitor host, run `etherfind` in RPC mode using the `-r` option. An example command appears below.

```
# etherfind -i interface -r -src clientname -o -dst clientname
```

The above command will likely capture a large amount of packet activity. It may not be immediately apparent where the problem is located, but this information can be forwarded to your local Sun service center for review. It can also be used for a comparison with a trace of a successful client boot case to help locate the failure point.

It is important to note debugging messages from the server `bootparamd` process if `etherfind` shows some client and server interaction. These debugging errors may highlight more subtle problems occurring on the server-side if YP or DNS or both services are used. Error messages suggesting YP- or DNS-related service problems include those shown below.

```
Whoami failed: gethostbyaddr for address
bp_getclntent failed
bp_getclntkey failed
```

If the client's server-side has a YP binding, then the server will make requests to a YP server to resolve `hostname` and `bootparams` information. Problems at the YP level (e.g. `domainname`, server not responding, bad map data) will prevent the client's boot process from succeeding and should produce a debug error as shown above.

If you suspect problems at the YP level, try configuring the client server with enough local file information (i.e. `/etc/hosts`, `/etc/bootparams`, `/etc/exports`) to support the client boot without YP. Then kill the `ypbind` process and reboot the client. If the client boots successfully without the server running `ypbind`, then you know YP-related problems need debugging.

Additional problems can be found at this YP level if the YP server has been configured to use DNS (Domain Name Service), causing client-`hostname`

resolution at the DNS level. Client boot processes may not succeed if DNS service is interrupted or anomalies with hostname matches exist.

If you suspect problems at the DNS level, you should make YP hosts maps with appropriate host information to allow the YP database to support hostname resolution and thus avoid DNS-related problems. Of course, the DNS-related problems should then be debugged and corrected.

Once stage 2 network failures have been identified and fixed, the client should be able to successful NFS-mount its remote `root` and `swap` filesystems and boot a `vmunix` (stage 3).

Stage 3 Debugging

Diskless client boot failures in stage 3 usually suggest problems with the `vmunix` itself, problems with filesystem paths, or problems with client-space system files or daemons. This, however, can be a problem on any type system during this state of booting a `vmunix`.

Generally, debugging at this level involves checking client-space filesystem links on the server-side, verifying a correct `vmunix` is being booted, and finally checking the client-space `rc` files for potential system initialization problems.

Start by checking that the server's `/etc/exports` file is exporting the correct filesystem paths with the correct access. Then verify that any links to a `vmunix` or `/usr` areas are correct. Finally, if you are using any common patches or custom programs that a client should also use, ensure that these modifications were also made to the appropriate client space.

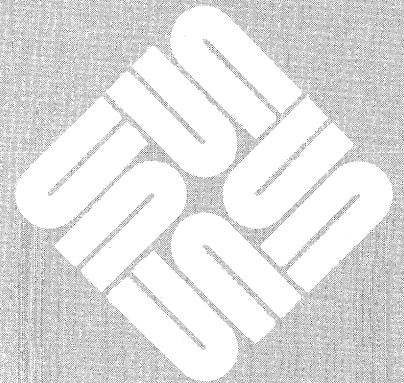
Known Bugs: 1018583,
1018791, and 1013639

Diskless clients running SunOS releases 4.0.x having boot PROMS less than revision 3.0 can hang and ultimately fail during boot in cases where multiple boot servers are responding, or an improperly-implemented portmapper exists on the network.

In these cases the client boot fails with the console often indicating an unusual sleep or exception message, or an RPC failure status. Patches are available through your local service center.

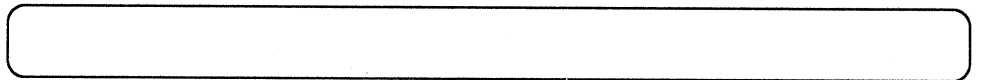
STB SHORT SUBJECTS

STB SHORT SUBJECTS	1419
Selection Service Messages	1419
valloc failed Defined	1420



STB SHORT SUBJECTS

Selection Service Messages



Selection Service Message Defined

Customers may receive the message shown below from the selection service which uses RPC calls.

```
Request to current holder failed: RPC: Timed out
```

This message may result from the network or CPU load on your machine, or from a race condition in the selection service. The race condition results when one window tries to notify the selection service that it has the primary selection while another window asks the selection service 'Is there a primary selection right now?'

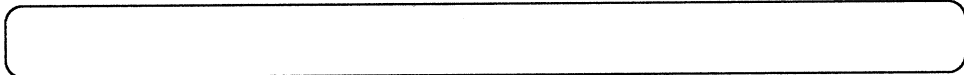
These two simultaneous requests collide, resulting in the time out of the RPC calls used by the selection service.

The Workaround

If the race condition is the problem, you can use the secondary selection instead of the primary selection. Press **(L8)** while selecting the text. You will see an underline instead of reverse video.

Please note that this message can also be a symptom of a network problem.

valloc failed **Defined**



valloc failed **Message Definition**

Customers trying unsuccessfully to get a window tool to come up may receive the message shown below.

```
pr_makefromfd error: valloc failed
pr_open: pixrect create failed for /dev/fb
```

The Problem **Defined**

The above message appears when there is insufficient swap space on the machine to start the tool.

For example, in order to paint the screen, `lockscreen` and any other SunView program maps the frame buffer into its address space. In order to do this, the program first allocates memory equal to the frame buffer size and then maps the frame buffer on top of the memory. Once the frame buffer is mapped, the allocated memory no longer requires any swap space.³ However, it *does* require swap space until it is mapped.

Customers seeing the above message have probably run out of swap space, so SunView programs cannot allocate memory on top of which to map the frame buffer. Swap space for data is allocated in sections that get larger as the data space gets larger. The swap space becomes fragmented, so that a contiguous section of memory large enough for the program may not be available, even when there are enough 'fragmented' spaces to make up the required size.

A Workaround and A Fix

Use the workaround or fix shown below, depending on how often this problem arises.

- Exit and reenter `suntools`.

This frees swap space, so the 'fragmented' sections of memory no longer have allocated sections among them. When you reenter `suntools`, you may now have enough contiguous swap space to run `lockscreen` or other SunView programs.

- Increase the swap space.

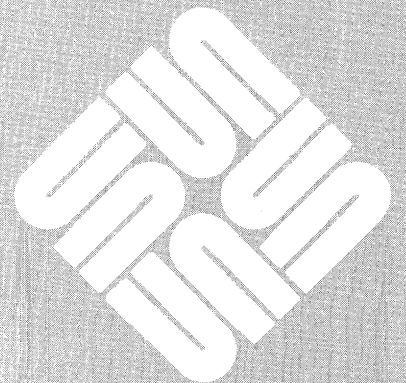
You may need to do this in case seeing the above message becomes a continual problem.

³ Note that the allocated memory will require swap space after mapping the frame buffer in the unlikely case you have mapped something else on top of the allocated memory.

IN DEPTH

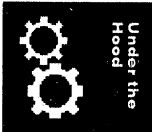
IN DEPTH 1423

Sun386i Administration Cookbook: Ch. 9, Under the Hood 1423



Chapter 9: Under the Hood

9.1 Introduction	Page 111
<hr/>	
9.2 Inside the Sun386i File System	Page 111
<hr/>	
9.3 Inside the Automounter	Page 119
<hr/>	
9.4 Secure RPC	Page 124
<hr/>	
9.5 Network Time Synchronization	Page 129
<hr/>	
9.6 Inside SNAP, ASI, and New User Accounts	Page 131



9.1 Introduction

This chapter discusses the Sun386i file system, the workings of Secure RPC, and the machinery of SNAP, Automatic System Installation, and New User Accounts.

9.2 Inside the Sun386i File System

If you are familiar with other Sun systems and have worked with Sun386i systems for any time at all, you know there are differences between the typical file system hierarchy on Sun-3, Sun-4, and SPARCstation systems and the file system as it is shipped on Sun386i disks. The Sun386i file system hierarchy was laid out differently to accommodate the needs of customers who:

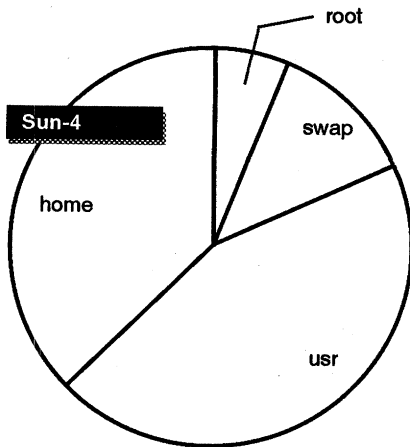
- ◆ Want a system that boots on delivery, with core software preloaded
- ◆ Do not want hard limits on specific file systems (for example, /tmp) which could force a repartition
- ◆ Require compatibility with standard SunOS directories (/usr/bin, /usr/local, etc.) so that scripts and procedures can run on all systems

The first step in accommodating these goals was to divide Sun386i disks differently from the way Sun installation software typically sets up disks on Sun-3, Sun-4, or SPARCstation systems. The second step was “redirecting” some standard SunOS files and directories to take advantage of the new disk layout.

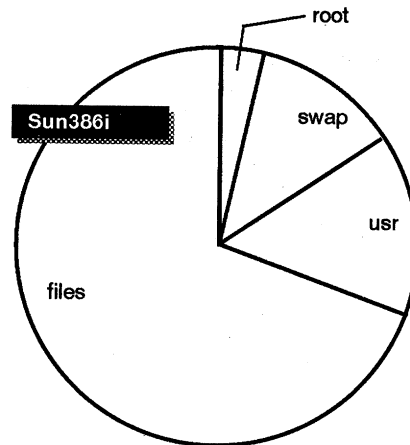
Comparing Partitions

On most Sun systems, users must choose where they will allocate available disk space; the default in `suninstall` is to split user-available disk space between two roughly equal partitions. On Sun386i systems, the bulk of user-available disk space is in one partition (/files), so that growing files can compete equally for free space. The Sun386i arrangement guarantees that users never have to repartition their disks to gain more usable file space.

Here is a comparison using two hypothetical system disks of 130 Mbytes:



Partitions on a typical Sun-4 client's disk



Partitions on a Sun386i disk

An explanation of these disk layouts is included on the next page.



root and swap – The default sizes of root and swap are approximately the same on both the Sun-4 and Sun386i disks.

`/usr` – The Sun386i `/usr` partition stores Sun-supplied software and other system files. This partition is almost completely full and is mounted read-only. On other Sun systems, a large amount of user-available space is left in `/usr`, where user files and third-party software are often stored. See “More About `/usr`” (page 117) for additional information on the Sun386i `/usr` partition.

`/files` and `/home` – On a Sun386i system disk, the `/files` partition accounts for almost all of the user-available space. This partition houses many different types of files, including user files and home directories, optional SunOS commands, and files that are likely to grow or shrink such as temporary files or mail.

On the Sun-4 disk, the additional partition is called `home` and most user file space is thus split between the `usr` and `home` partitions.

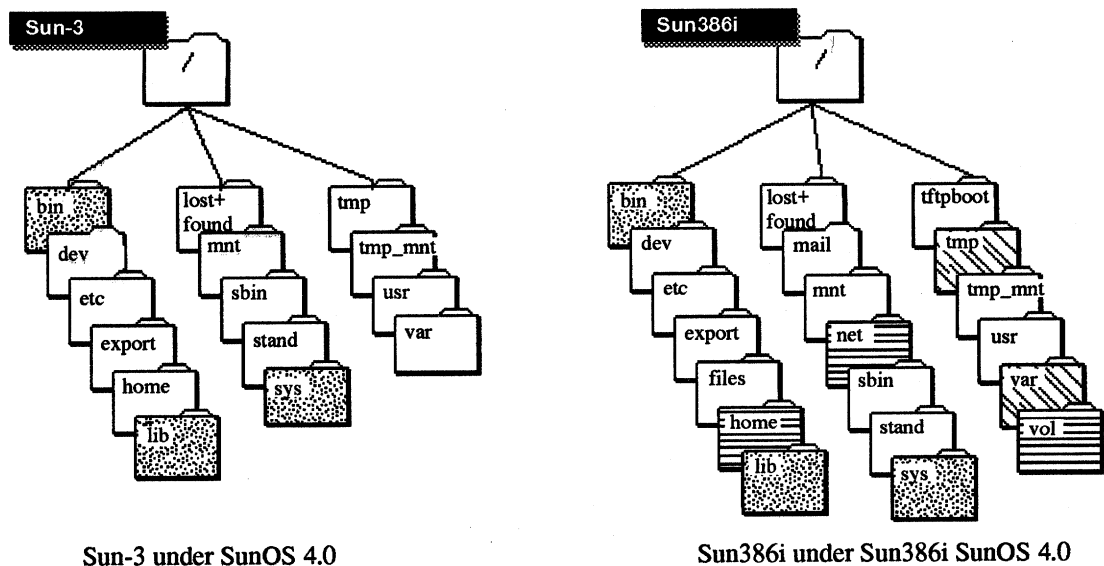
An additional and sometimes confusing point is that Sun386i systems use `/home` as an automount point for home directories. Other Sun systems do not use this automounting convention by default. See “Inside the Automounter” for details.

Compatibility between Directory Structures





Although the Sun386i disk layout scheme solves many of a user’s potential repartitioning headaches, compatibility with existing SunOS file and directory locations was also required. For example, even though optional commands such as `spell` are stored in the Sun386i `/files` partition, users need them to appear in traditional locations such as `/usr/bin` so that existing search paths, shell scripts, and ported software can run across all Sun platforms. And even though space for files that grow or shrink is allocated to the `files` partition, users expect to see such files in standard SunOS 4.0 directories such as `/var` and `/tmp`.

The Sun386i file system thus “redirects” many traditional SunOS directories using standard SunOS features such as symbolic links and loopback mounts (new in SunOS 4.0).

The figure below shows some of the differences between the file system layout found on the Sun386i and other Sun systems:



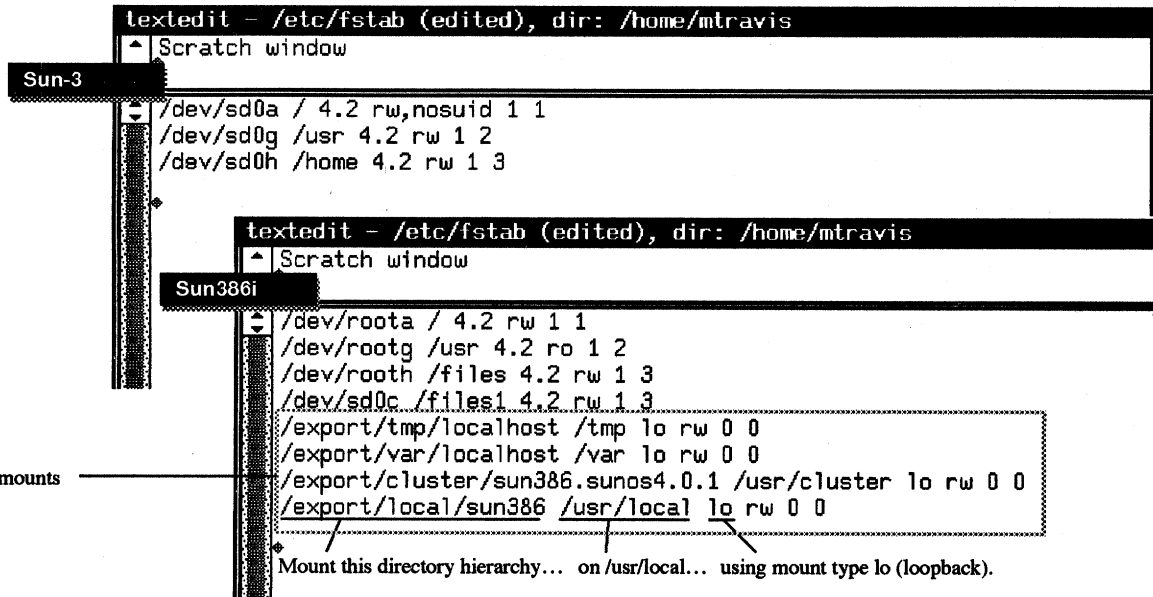
(See the next page for an explanation of the directory types shown.)

-  Real directories – These are standard UFS directories.
-  Symbolic links – All SunOS 4.0 file system layouts use symbolic links to maintain compatibility with older directory structures. A Sun386i system also uses additional symbolic links in lower directories (not shown here) for other reasons explained later in this section.
-  Loopback mounts – These are special SunOS 4.0 mounts that perform a function similar to that of NFS mounts, except that they are not remote. Through loopback mounts, standard SunOS directories such as /tmp appear to users in their normal location (in /), even though the actual files are located in the Sun386i /files partition where more space is available. See “About Loopback File Systems” for details.
-  Automount points – On Sun386i systems, files in /home, /vol, and /net are automatically mounted as needed. The automounter is available on all SunOS 4.0 systems, but only Sun386i systems activate it by default.

About Loopback File Systems

The loopback file system is a new file system type available under all releases of SunOS 4.0. With a loopback file system, you can mount a local directory hierarchy on top of another local directory. This functionality is the local equivalent to mounting a remote directory using NFS.

You can see how loopback mounts are used by comparing sample /etc/fstab files from Sun-3 and Sun386i systems. The default Sun-3 /etc/fstab file contains no loopback mounts, while the Sun386i /etc/fstab file has four:



```

textedit - /etc/fstab (edited), dir: /home/mtravis
Sun-3
/dev/sd0a / 4.2 rw,nosuid 1 1
/dev/sd0g /usr 4.2 rw 1 2
/dev/sd0h /home 4.2 rw 1 3

textedit - /etc/fstab (edited), dir: /home/mtravis
Sun386i
/dev/roota / 4.2 rw 1 1
/dev/rootg /usr 4.2 ro 1 2
/dev/rootg /files 4.2 rw 1 3
/dev/sd0c /files1 4.2 rw 1 3
/export/tmp/localhost /tmp lo rw 0 0
/export/var/localhost /var lo rw 0 0
/export/cluster/sun386.sunos4.0.1 /usr/cluster lo rw 0 0
/export/local/sun386 /usr/local lo rw 0 0
    
```

Mount this directory hierarchy... on /usr/local... using mount type lo (loopback).



Directories containing files that are likely to grow in size (`/var` or `/tmp`, for example) are set up in a file system where free space is available, and they are then loopback mounted onto the standard locations. Ultimately, these mounts resolve to subdirectories in `/files` (usually via symbolic links in `/export`), where they compete equally for available space in the partition.

For such directories, loopback mounts offer the best of both worlds: Directories such as `/tmp` appear in their standard locations to a user, and yet really point to a partition where more ample free space is available. Traditional UNIX chores such as having to guess the high-water mark for directories like `/tmp` are no longer necessary.

Although symbolic links might have been used for this kind of redirection, loopback mounts are preferable for two primary reasons:

- ◆ Mount points for loopback mounts (for example, `/var` or `/tmp`) are real directories. They can still be used in the event that nothing is loopback mounted on top of them, as is the case when a system boots in single-user mode.
- ◆ Loopback mounts are specific to a system. There is always a separate `/etc/fstab` file for every workstation since the root file system for each workstation is unique. In contrast, a symbolic link could be in a file system that is shared by many workstations—potentially limiting file system flexibility.

The Sun386i file system layout attempts to make optimum use of either loopback mounts or symbolic links, whichever is more appropriate in a given situation. Additional information on symbolic links is contained on the following pages. Also see “About `/export`” (page 114) for information on Sun386i exporting conventions.

Troubleshooting Loopback Mounts

Here are some common problems with loopback file systems, and their solutions:

mount_lo:No such file or directory – One of the two directories specified in the loopback mount line does not exist. Often this is because the directory is being mounted after the loopback mount entry in `/etc/fstab`. A loopback mount entry in `/etc/fstab` must be placed after the mount points of both directories it depends on. This is most easily accomplished by placing the loopback mount entries at the end of `/etc/fstab`.

mount_lo:No such device – Whenever you rebuild a Sun386i kernel, you must include the loopback file system in the kernel configuration, using the `LOFS` option. This line is included in the preconfigured kernel.

mount:Unknown file system type:lo – When the `mount(8)` command encounters a file system type that isn't built in, it starts a program called `mount_filesystem_type`. In the case of loopback mounts, this is `mount_lo`, which exists in `/usr/etc`; in the event of problems with mounting loopback file systems, check that `/usr/etc` is in the path. (Note that the standard `/etc/rc*` files on Sun386i systems set the path correctly.)

Because `lo` is simply another file system type, you can use it with any command that takes an `fstype` argument. For example, here's how to use `df` to get a list of mounted loopback file systems:

```
{system:1} df -t lo
```

About `/export`

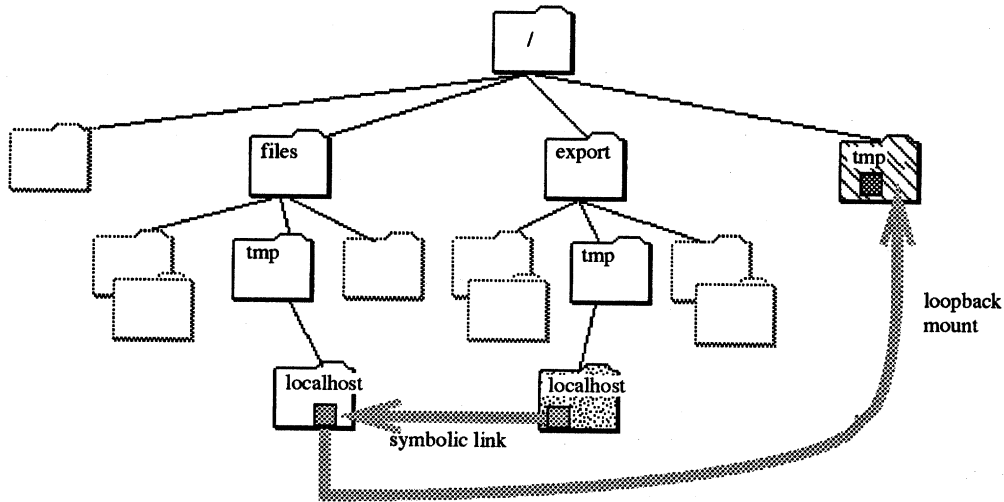
On all Sun systems including the Sun386i, the `/export` directory is intended as a convenient organizing place for directories that will be exported to other systems. For example, `/export/root/client` is a standard SunOS 4.0 exporting convention on the boot servers of diskless clients.

On Sun386i systems, symbolic links are used by convention to export file systems: The standard “directories” in `/export` are actually symbolic links to other partitions (usually `/files`).

This scheme allows a central and standard place for exported files, while remaining true to the goal of keeping most available free space in the `/files` partition. Customers are also encouraged to create symbolic links from `/export` to any new partitions they add (`/files1`, for example), and to use these symbolic links when exporting the new partition.

Some of the symbolic links in `/export` resolve to directory hierarchies, which are in turn loopback mounted onto other local directories.

For example, `/export/tmp/localhost` is a symbolic link to the directory `../../../../files/tmp/localhost`, which is in turn loopback mounted onto `/tmp`.



While this combination of loopback mounts and symbolic links is confusing if you are trying to trace a particular directory to its source, the scheme can ultimately simplify administration of exported directories and disks.

About Relative (..) Symbolic Links

If you ever use `ls -l` to examine directory symbolic links in the Sun386i file system, you'll probably notice path names such as `../../../../files/tmp/localhost`. The `..`'s in such path names are standard SunOS notation for a parent directory. They make this a relative symbolic link (as opposed to an absolute symbolic link, which specifies a path name starting with `/`), ensuring that the contents of the symbolic link can be properly evaluated even when examining a diskless client machine's file systems from the server machine.

Mounting of Symbolic Links

On all Sun systems, symbolic links are followed at mount time. When a mount request is made for a directory such as `/export/tmp/localhost`, the symbolic link `/export/tmp/localhost` resolves to `/files/tmp/localhost` before the mount completes. Thus, the system actually mounts the directory `/files/tmp/localhost`.

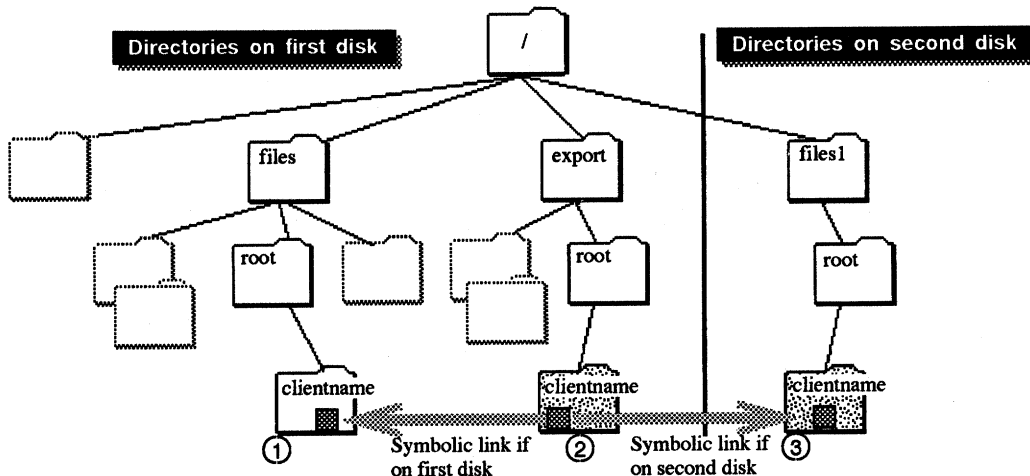
Links in Action: Adding Extra Disks Through `/export`

The convention of mounting symbolic links in `/export` is particularly effective for managing file systems on NFS servers.

Moving existing directories from `/files` on a server to the new disk `/files1` is relatively simple. NFS clients mount their files via the symbolic links in `/export`, so the only change required is to move the files and then make the relevant symbolic links point to the new partition on the server. No change is required on any of the client systems.



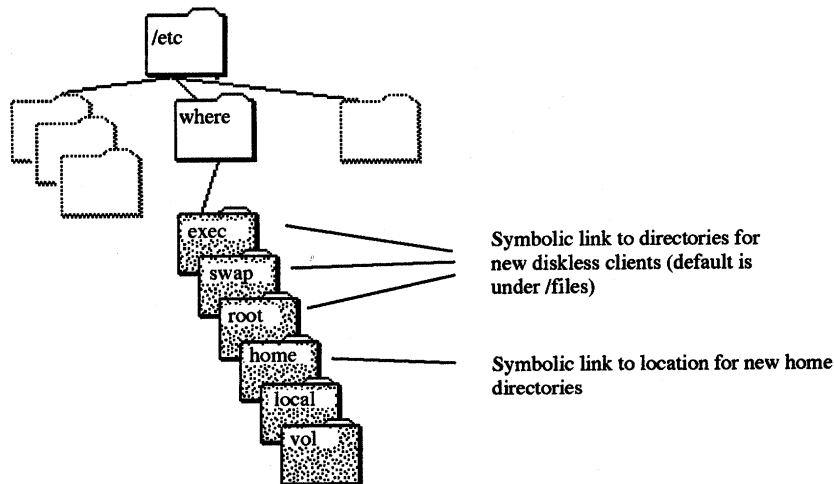
For example, if a diskless client's root were stored on /files, it would be in the directory /files/root/client_name (①) and there would be a symbolic link from /export/root/client_name (②) to it. The contents of the directory /files/root/client_name are moved to /files1/root/client_name (③) and /export/root/client_name is made to point to this new location.



Similarly, symbolic links for home directories are /export/home/groupname/username, and point to /files/home/groupname/username. These symbolic links enable you to move home directories in a similar way, and enable home directories to be split over multiple disks.

Additional Disk and the Use of /etc/where

Once an extra disk is added to a system, it is possible to indicate to various daemons that the new disk is the one to use for the creation of new home directories or for support of new diskless systems. This is done using the symbolic links in the directory /etc/where.



The symbolic links for exec, root, and swap point to the directory in which to create a diskless client's exec, root, and swap directories. By default these point to /files/exec, /files/root, and /files/swap.

By modifying these to point to, for example, `/files1/root`, `/files1/exec`, and `/files1/swap`, the corresponding directories for new diskless clients added to the system via SNAP or Automatic System Installation will be created on the `/files1` disk in the indicated directories. You can also modify just one link (for instance, `/etc/where/swap`), so that new swap areas for diskless clients are created on one disk and their `root` and `exec` areas remain on another.

Similar links called `home`, `local`, and `vol` exist in `/etc/where` for home directories configured by SNAP or New User Accounts, and also for local software and optional software available network wide.

More About `/usr`

The `/usr` partition on Sun386i systems is read-only, and contains only a core set of standard SunOS files. One benefit of this policy is that it discourages users from storing their own files in `/usr`, so that the `/usr` partition never fills up and requires repartitioning.

Other advantages to `/usr` being read-only include:

- ◆ Consistency among clients with and without disks
- ◆ Increased stability and less chance of corruption to the `/usr` partition
- ◆ Reduced boot time, since there is no need to run `fsck` on `/usr`
- ◆ Trouble-free sharing of `/usr`, since it is unlikely that `/usr` will have been modified or customized.

As of SunOS 4.0, the `/usr` partition on all Sun systems has been set up to be architecture-specific; all files that are not architecture-specific have been moved to other partitions. This means that it is easy to share `/usr` between systems of the same architecture. On other Sun systems, the `/usr` partition is exported read-only so that systems that remotely mount `/usr` (such as diskless clients) do not have write access to the `/usr` partition. Sun386i systems take this one step further by mounting `/usr` read-only on both diskful and diskless systems, so that `/usr` is consistent between clients with and without disks.

Third-Party Software and `/usr`

One disadvantage to the approach taken with `/usr` on Sun386i systems is that third-party software may require you to install files in the `/usr` partition. Usually, you can install such software in the `/files` partition, and set up a symbolic link or mount point to access the correct directory in `/files`.

For more information, see “Installing Third-Party Software” in Chapter 5.

Sharing `/usr` Among Clients

A Sun386i client that has a disk, but shares `/usr` with another system, is called a diskful client. Sun386i systems have no automatic support for sharing `/usr` among clients with disks; however, Appendix C of Sun386i SNAP Administration explains how to accomplish this easily. Sharing `/usr` among clients with disks conserves disk space across the network, but has some performance implications because it creates more network traffic and a heavier load on the server where `/usr` resides.

Performance of `/usr`

Even though `/usr` is more than 100% full, its read-only status renders it immune to the performance penalties that arise when allocating disk space on very full file systems. Because `/usr` is read-only, new blocks are typically not allocated in this file system and thus there is no impact on performance in this partition.



Optional Clusters

With Sun386i SunOS 4.0.2, all SunOS software is preloaded on the hard disk that is shipped with new Sun386i systems.

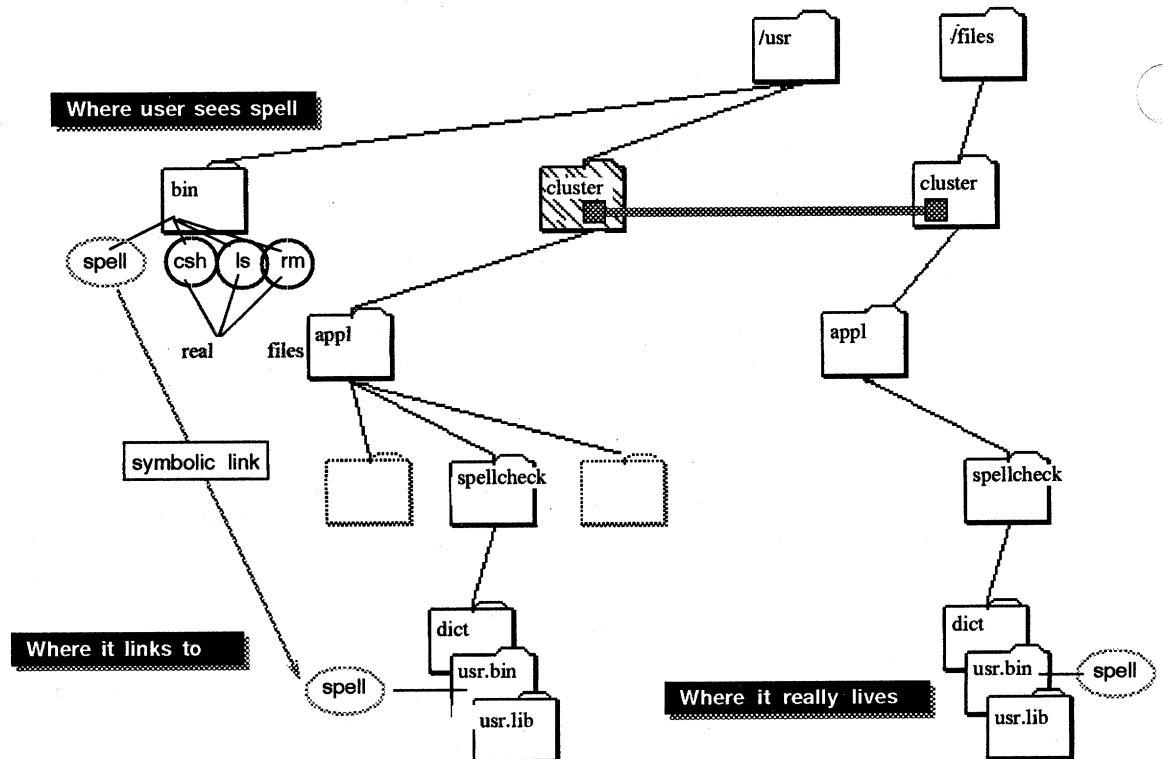
To accommodate both the disk size and the desire to leave as much space free on the disk as possible, customers can unload all but the most critical pieces of SunOS, called Application SunOS core system. The rest of the operating system is divided into groups of files called optional clusters, which are available on separate media. Customers can unload or reload only those portions of SunOS that they need, thereby retaining more disk space for user or application files.

(Note that when existing Sun386i systems are upgraded from 4.0.1 to 4.0.2, no additional clusters are loaded. Systems upgraded to 4.0.2 have the same clusters—in an upgraded form—that they had prior to upgrading.)

All Those Symbolic Links

To maintain compatibility with existing SunOS file system structures, every file in every Sun386i optional cluster has a symbolic link from its familiar file system location to its location on Sun386i systems. While the use of symbolic links may at times seem confusing, these links allow scripts and ported software to work identically across the Sun product line. After loading the appropriate cluster, you can access files within clusters using the same path as on other Sun systems.

For instance, the `spell` command is part of the loadable `spell_check` cluster. To a casual user, `spell` appears in `/usr/bin`. However, `/usr/bin/spell` is actually a symbolic link that points to `/usr/cluster/appl/spellcheck/usr.bin/spell`:



(Note that all clusters really reside in `/files`, which has the most free space; the `/usr/cluster` is actually a loopback mount that ultimately resolves to a directory structure in `/files`. See “About Loopback File Systems” and “About `/export`” on pages 113 and 114.)

Saving Disk Space with Clusters

In a networked environment, users can mount the optional clusters via NFS from a cluster server, saving even more disk space network wide. (See “Establishing Cluster Servers” in Chapter 6.) Sharing the clusters among clients with disks saves disk space on the network as a whole; but sharing clusters also has performance implications because it creates more network traffic and a heavier load on the cluster server.

9.3 Inside the Automounter

The automounter is a standard SunOS 4.0 feature that provides automatic mounting of a file system upon first access. It is an NFS server designed to simplify access to remote file systems. With the automounter running, a user who wants to access exported files on other systems no longer needs to know the superuser password, or to understand the workings of `/etc/fstab` or `mount(8)`.

The automounter also makes life easier for administrators, who can move entire directory trees from one file server to another without having to update individual systems’ `fstab` files. Instead, mount points are defined in centrally accessible automounter maps. (If YP isn’t running, these maps can be distributed to individual machines by a mechanism such as `cron(8)`.)

The automounter can be used in addition to older mounting techniques: Mounting file hierarchies with the automounter doesn’t preclude the use of `mount(8)` to mount hierarchies onto other directories in the traditional way.

What the User Sees

The casual user does not need to know the conventions of the automounter, and in fact does not even need to be aware that it is running.

To take advantage of the automounter, a user simply changes to the automounted directory, or refers to the files directly via their path names—just as he or she would with any other file or directory. Here is a sample session with an automounted home directory:

```
{system:1} cd /home/mtravis
{system:2} ls
daily.articles/  weekly.articles/  monthly.articles/
{system:3} ls weekly.articles
WSJ      Barrons  Fortune
{system:4} ls /home/mtravis/weekly.articles
WSJ      Barrons  Fortune
```

For a technical explanation of what actually happens when you refer to a file or directory via the automounter, see Appendix A.



Standard Sun386i Automounted Directories

The three standard automounted directories on Sun386i networks are:

- ◆ **/home** – The path to users' home directories, accessed as `/home/username`. Users can log into any machine running the automounter and be in their home directories.
- ◆ **/vol** – Used for additional network-wide directories that a site administrator wants to set up.
- ◆ **/net** – Used to access workstations on the network, as `/net/hostname/pathname`. For performance reasons, `/vol` or `/home` is preferred over `/net`. However, `/net` can be useful for getting data from infrequently accessed machines whose files aren't available through `/home` or `/vol`.

Directories under `/home`, `/vol`, and `/net` are mounted only on first access. For instance, if you type `ls /home/mtravis`, the automounter mounts only `mtravis`' home directory.

To avoid a gradual buildup of remote mounts, the automounter unmounts file systems that have not been used for five minutes or more.

How the Automounter Starts

On Sun386i systems, the automounter is started at boot time by a set of lines in `/etc/rc.local`. On other Sun workstations, users who want to use the automounter must start it themselves (as superuser) or add a similar line to `rc.local`.

You can see how the automounter starts by examining these `rc.local` lines:

```

textedit - /etc/rc.local
Scratch window
if [ -f /usr/etc/automount ]; then
  find /tmp_mnt/* -depth -xdev -type d -exec rmdir {} \;
  if ypmatch /vol auto.master > /dev/null 2>&1
  then
    automount -tw 300 && chat -n ' automount'
  elif [ -f /etc/auto.vol ]; then
    automount -tw 300 -m /net -hosts /vol /etc/auto.vol && chat -n '
automount'
  fi
fi
if [ -f /usr/etc/in.rwhod -a -d /var/spool/rwho ]; then
  in.rwhod &
  chat -n ' rwhod'
fi

```

Check for YP automount maps

Start automounter using the auto.master YP map

Start automounter using local files

Check for YP automount map – This line checks to see if a `/vol` entry exists in the YP `auto.master` map. It will pass this test only if YP is running and there is an `auto.vol` entry in that map.

Start automounter using YP maps – If the YP `/vol` entry exists, the automounter starts using the default YP `auto.master` map. The `-tw 300` entry sets a time-out of 300 seconds (five minutes) after which automounted file systems will be unmounted. If verbose boot messages are enabled (see Chapter 4), then the message `automount` is displayed on the screen.

Start automounter using local files – If YP isn't running, or there's no `auto.vol` entry in the `auto.master` map, then this line tries to automount `/net` from the local `/etc/hosts` file, and to automount `/vol` from the local `/etc/auto.vol` file.

What's in the Automounter Maps

By default, a Sun386i system uses four standard automounter maps: `auto.master`, `auto.home`, `auto.vol`, and the special built-in map for `hosts`.

The master Map

The `auto.master` map contains entries for the various automount points and names the automount maps that will be used to manage those mount points. Unless you specify otherwise, the automounter attempts to read this map when it starts. The default Sun386i `auto.master` map includes entries that define the maps for `/home`, `/vol`, and `/net`:

```

textedit - /etc/auto.master, dir: /etc
Scratch window
# @(#)auto.master      1.6 Copyright 1987 Sun Microsystems Inc.
#
# DIRECTORY      MAP      MOUNT OPTIONS
/net             -hosts      -intr,nosuid
/home           auto.home   -intr,nosuid
/vol            auto.vol    -intr,nosuid
    
```

Mount options work like those in `/etc/fstab`

Note that the default options for all three file systems specify `nosuid`. This practice eliminates the possibility of a user gaining superuser privileges on a variety of hosts via the automounter. If an `suid` mount option is necessary, specify it in the mount options of a particular map entry in `auto.vol` or `auto.home`, and this will override the default set up in the `auto.master` map.

Home Directories and auto.home

The `auto.home` map lists the mount points for users' home directories. This file is updated when user accounts are created through SNAP and New User Accounts, and also when SNAP is used to move or delete user accounts. As you can see, the automounter is capable of mounting home directories from any home directory server:

```

textedit - /etc/auto.home (edited), dir: /etc
Scratch window
# key mount-options location
#
mtravis ← oak:/export/home/users/mtravis
juser ← grumpy:/export/home/users/juser
ahinkle ← gcs:/u/accts/ahinkle
    
```

Accessed as...

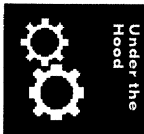
```

/home/ahinkle
/home/juser
/home/mtravis
    
```

Sun386i home directories exported from oak and grumpy via `/export`

Home directory on system gcs, stored in a directory called `/u`

On Sun386i home directory servers, home directories generally reside in `/files/home` or `/files1/home` (the expansion unit disk) but the convention is to mount them via symbolic links in `/export/home` as shown in the example.



Network-Accessible Directories and `auto.vol`

The `/vol` hierarchy is useful for automounting "volumes" such as groups of DOS applications, source code, or third-party software. `/vol` is a Sun386i convention.

As shown in the example below, the `auto.vol` map lists the mount points for `/vol`:

```

textedit - /etc/auto.vol (edited), dir: /home/mtravis
Scratch window
# key mount-option location comment
#
help -ro cleo:/usr/hi/help # HELP, juser
help.master cleo:/usr/hi/help # HELP, juser
archives geppetto:/export/archive # archives, MIS
dosapps pinocchio:/export/apps/dos # dos stuff, mtravis
Accessed as...
/vol/dosapps
/vol/archives
/vol/help.master
/vol/help
Mount options here override those in auto.master
Comments are useful for tracking use and identifying local administrator

```

As shown in this example for the `/vol/help` entry (which is mounted read-only), you can specify mount options to override the default mount settings.

Automounting from Non-Sun386i Home Directory Servers

On Sun-3, Sun-4, and SPARCstation systems running SunOS 4.0, conventionally `/home` is a UFS mount point and users' home directories are specified as:

```
/home/servername/username
```

To automount a such a home directory:

- ◆ Ensure that the `passwd` map specifies `/home/username` in the home directory field.
- ◆ Use the following format in the `auto.home` map:

```
username servername : /home/servername/username
```

The above steps will give access to this home directory on all machines that run the automounter using that `auto.home` map. You will need to perform the following additional steps on machines that do not run the automounter:

- ◆ Mount the home directory via `mount(8)` and `fstab(5)`. You can mount it on `/home/servername/username`, or in another convenient location.
- ◆ Create a symbolic link from `/home/username` to where the home directory is mounted

With this method, you are using the Sun386i convention for the home directory name across all your systems.

Running the Automounter on Non-Sun386i Home Directory Servers

The above procedure works in most cases, but won't work if you run the automounter with the `auto.home` map on servers where the `/home/servername/username` directory actually resides. An alternate solution is to rename all `/home` partitions in 4.0 servers to `/files` or some other designation of your choosing. The `auto.home` entry for `ahinkle` would then look like this:

```
ahinkle servername:/files/servername/ahinkle
```

By renaming the `/home` partitions on home directory servers, the automounter can be run on all systems—including home directory servers—without conflicting with existing UFS mount points.

If you rename home directories in this way, remember also to change the home directory field for each affected user in the `passwd` maps.

Again, this method is using the Sun386i convention for the home directory name across all your systems.

Troubleshooting the Automounter

The following are frequently encountered problems and their solutions.

Can't access all files on a remote system – The automounter mounts only exported file systems. So, certain directories that you can see by remotely logging in to a system typically won't be available.

Solution: Either export the file system you need to access, or use commands such as `rcp` and `rsh` to access the files directly.

`/net` doesn't show any files on some systems – The automounter cannot access files on diskless systems or on systems that are not NFS servers. This is because such systems do not export their files. For example, you can't use the automounter to access files and directories on a PC running PC-NFS.

Slow performance of `/net` – `/net` mounts every exported file system when accessing a given server, so it is slower than `/vol`. Use a `/vol` path if it is available.

Unusual path names such as `/tmp_mnt/home/mtravis` – On all Sun platforms, the automounter shows symbolic links to temporary mount points when you use `pwd` or other commands. By default, Sun386i systems hide this by setting the `AUTOMOUNT_FIXNAMES` environment variable to `TRUE` in new users' `.login` files. The `AUTOMOUNT_FIXNAMES` setting is used by `getwd()` to prevent the `/tmp_mnt` prefix from showing. If you notice unusual automounter paths, check to make sure that `AUTOMOUNT_FIXNAMES` is set. `AUTOMOUNT_FIXNAMES` is a Sun386i-only feature.

Can't find file that it previously opened – Because the automounter periodically unmounts idle file systems, certain programs may be unable to re-access files that were opened via a `/tmp_mnt`-style path name.

Solution: Make sure `AUTOMOUNT_FIXNAMES` is set to `TRUE` on Sun386i systems (see above).

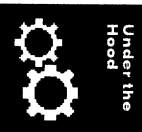
Doesn't recognize new maps added to `auto.master` – In certain cases, you may want to add new automounter maps by setting up additional entries in `auto.master`, editing the `YP Makefile` and then remaking YP. The section on "Setting Up a Cluster Server" in Chapter 6 shows an example of how to do this.

You can modify the automounter maps at any time, but keep in mind that the automounter only looks at the master map when you run the `automount(8)` command, which is normally started at bootup on Sun386i systems (when they read `/etc/rc.local`). Therefore, to use a map modification immediately, restart the automounter on each system that needs to take advantage of the new map.

hostname:filesystem already mounted on mountpoint – The automounter has mounted a file system on a directory that already had a file system mounted on it. This happens if an entry in an automounter map also appears in a system's `/etc/fstab` file (either by accident or because the output of `mount -p` was redirected to `fstab`).

Solution: Delete one of the redundant entries.

trymany: servers not responding: reason – No server in a replicated mount list is responding. This may indicate a network problem, or that a 4.0.1 automounter is running with



a map that specifies replicated server entries. (Replicated server entries were not supported in Sun386i SunOS 4.0.1.)

Remount hostname:filesystem on mountpoint: server not responding – The automounter attempted a remount after an unmount failed. Indicates a server problem.

NFS server (pidn@mountpoint) not responding still trying – An NFS request made to the automount daemon with process ID n serving mountpoint has timed out. The automounter may be temporarily overloaded or not active.

Solution: Wait a few minutes. If the condition persists, the easiest solution is to reboot the client.



Diskless clients and the automounter – Diskless clients must always mount their root, swap, and usr partitions via mount(8) even if they use the automounter to mount some file systems.

9.4 Secure RPC

All SunOS 4.0 networks running YP provide the Secure RPC facility, a new feature of remote procedure calls that includes a mechanism for secure authentication of users and systems. Secure RPC enables servers to verify the identity of their clients.

The Secure RPC facility works exactly the same on all Sun workstations.

Services that use Secure RPC verify the identity of their clients before executing the requested procedure on behalf of those clients. On Sun386i systems, for example, the IP address allocation daemon is contacted by both SNAP and Automatic System Installation (ASI). The `rpc.ipallogcd` service validates the user invoking SNAP or ASI before performing the requested IP address allocation.

Services using Secure RPC include:

- ◆ Sun386i agents and allocator daemons that SNAP, ASI, and New User Accounts use (`rpc.uid_allogcd`, `rpc.gid_allogcd`, `rpc.ipallogcd`, `rpc.pnpd`, and `rpc.user_agentd`)
- ◆ YP updating (`yppatchupd`), except for the `yppasswd(1)` program

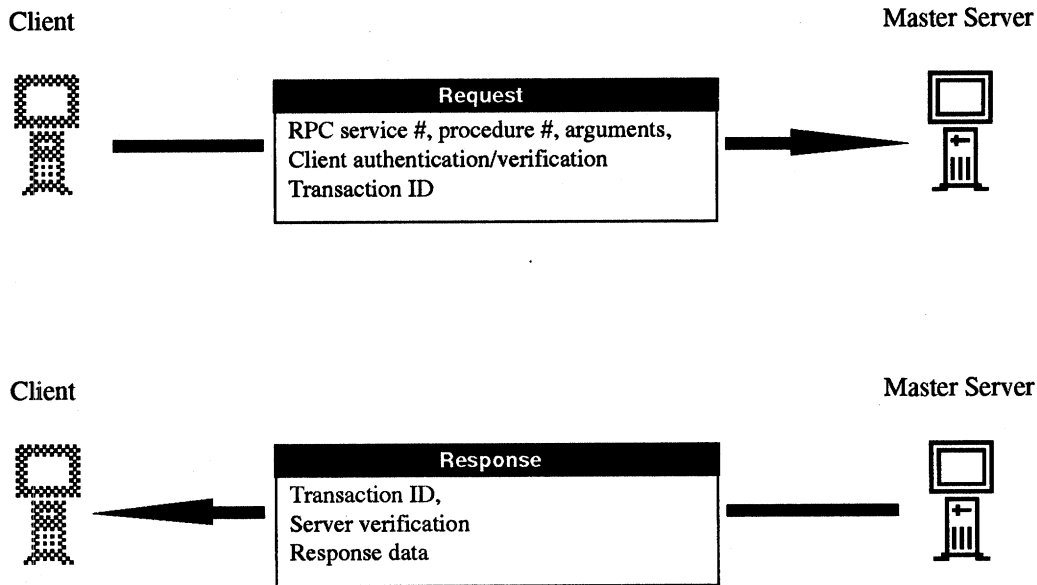
Any user accessing one of these services is validated with the default key (`nobody`) shipped with the system, or with a unique key if an administrator or user has created one with the `newkey(8)` or `chkey(1)` command. Similarly, superuser contact from a system to one of these services is validated with the default key, or with a unique key. If a user or the superuser on a system does not have a unique key, these Secure RPC requests use the default (`nobody`) key. If the default key is deleted from the system, then users who rely on the default key are unable to access any services using Secure RPC. The “Keys and YP Maps” section provides more information.

- D** By default, when a system is added via ASI, a unique key is set up for root of that system. Conversely, when a system is added via SNAP, no unique key is set up for root.

Neither SNAP nor the New User Accounts feature creates a unique key for a new user when setting up the account.

RPC Authentication

The basic RPC mechanism is a request/response protocol; the client sends a request to the server, and receives a response. The requests include information describing the request (RPC service number, procedure number, arguments), the requestor (authentication and verification information), and a transaction ID. The response includes the transaction ID used with the request (to distinguish between responses), information used to verify that the server is really the right server, and the response data.



Keys and YP Maps

Programs that use Secure RPC to communicate with other processes validate users by checking the `publickey.byname` YP map. This map is based on the contents of the file `/etc/publickey` on the YP master. The `publickey.byname` map contains an entry for each user and system on a network if (and only if) a unique key has been set up for that user or system. A system entry is used as an entry for that machine's root, so the map can contain an entry for each regular user and each root user (one per system) on the network. Clients and servers using Secure RPC generate "conversation" keys based on `publickey.byname` entries.

Each map entry in `publickey.byname` consists of a network name and a public and private key. Network names, also known as "net names," have the following format:

- `unix.userID@YP_domain_name` for users
- `unix.system_name@YP_domain_name` for superuser (root)

Each private key is DES encrypted with the user's (or root's) password. Each public key is generated from the private key. By default, the password used to encrypt the private key is the same as the login password.

Because of the encryption scheme, you should only add entries to the `publickey` maps with the `chkey(1)` or `newkey(8)` commands. Do not edit the `/etc/publickey` file by hand except to delete entries or to re-enter the default key as described in "Public Key Manipulation and Storage" on page 128.



nobody Default Key

Each Sun386i YP master contains a default key, called `nobody`, in the `publickey.byname` map. The map contains other entries only if:

- ◆ A system has been installed with ASI, in which case `publickey.byname` has a unique entry for the superuser (root) on that system
- ◆ A user or administrator creates a unique key with the `chkey(1)` or `newkey(8)` command for a given user

If no unique key is assigned to a user or root, the `nobody` key is used instead. The `nobody` key offers less security than individually assigned keys since the all-null password for `nobody` is well known; it is possible for clients or servers to impersonate other machines and issue or accept secure RPC calls.

If the entry for `nobody` is changed or deleted, users or root without their own entries in the `publickey.byname` map will be unable to communicate with processes that use Secure RPC. If you delete the `nobody` entry, you can retrieve it by entering:

```
system: SUPERUSER:1} grep nobody /usr/etc/unconfigure
```

Then add the entry to `/etc/publickey` on the YP master and remake the map (`cd /var/yp; make`). This is the only entry you should ever add by hand to the `/etc/publickey` file, since the password for `nobody` is unencrypted (all nulls).

netid.byname Map

Secure RPC also uses the `netid.byname` YP map, which contains privileges generated from the password, group, and host YP maps. Privileges that are stored in `netid.byname` can be of two types:

- ◆ A UID and a list of GIDs; for example, `1037:12,260,4,14,42,37,2359`
- ◆ Superuser for a host; for example, `0:helium.East.Sun.COM` (If you log in as root and issue a Secure RPC, your net name will be the host name.)

By default, only the net names of users and superusers of systems in the local YP domain are listed in this map, but requests from other domains can be authenticated and granted privileges (see Chapter 8, which discusses administering multiple domains).

The keyserv Daemon

The `keyserv` daemon must know your decrypted private key so that a conversation key can be created when you contact a service using Secure RPC.

When a user logs in, `login(1)` or `logintool(8)` decrypts the private key and password and then gives the result to the `keyserv(8C)` daemon. The `keyserv` daemon stores information on this in the `/etc/keystore` file. It also stores information in the `/etc/keystore` file if a user enters the `keylogin(1)` command with the correct password necessary to decrypt the user's private key.

You can give the private key information for root to the `keyserv` daemon by first killing the `keyserv` daemon, and then invoking the `keyserv -n` command. This command creates the `/etc/.rootkey` file, which stores the private key for root. The `/etc/.rootkey` file enables root on a system to use Secure RPC prior to the validation that occurs at login time (this feature is used by the daemons).

Troubleshooting Secure RPC

Secure RPC is transparent to users, but there are failure modes that are visible to users. These failures are usually due to mistakes in manipulating the new publickey databases used by Secure RPC; the details of those databases have not been well documented.

As delineated in this section, most Secure RPC problems can be corrected by performing both of the following steps:

- ◆ Deleting keys for users and root (except nobody) from the `/etc/publickey` file in multiuser mode and rebuilding YP
- ◆ Deleting the `/etc/keystore` and `/etc/.rootkey` files in single-user mode and rebooting

You can then recreate keys with the `chkey(1)` or `newkey(8)` command, and have users log out and back in again to reregister their keys with the `keyser` daemon. If you don't re-create the unique keys and register them after deleting nondefault keys and the `keystore` and `.rootkey` files, everyone will use the `nobody` key.

Authentication problems while running applications — If a network administrator deletes a user's entry from the `/etc/publickey` file on the YP master and rebuilds YP, or if a user changes it with the `chkey(1)` command, various RPC applications (such as secure NFS or SNAP) might generate errors with RPC credentials or verifiers. The problem will often go away if, in single-user mode, you delete the file `/etc/keystore` from the system where the user logged in, and then reboot.

After invoking `/usr/etc/chkey` to change the keys for a user or root (or deleting keys by editing `/etc/publickey`), make sure the right private key is in use by having the user (or root) log out and log back in again. As an alternative, users can change the private key in use by running `/bin/keylogin`.

- ✓ **keylogin(1) and the Domestic Kit** – In Sun386i SunOS 4.0.1, a bug requires that you install the Domestic Kit if you want to use `keylogin(1)`. This bug is fixed in 4.0.2.

Authentication problems with systems (root) — Similarly, you could see RPC errors after deleting the public key entry for root from the `/etc/publickey` file. To avoid these error messages after deleting a system's root public key entry, in single-user mode delete the `/etc/.rootkey` file, which stores root's private key, and the `/etc/keystore` file. Then reboot the system.

User or administrator forgets public key password — If you can't remember the correct public key password for an account, you can set the password and assign a new public key database entry by logging in to the YP master as root and invoking `/usr/etc/yp/newkey` for the user (or root). Note that on Sun-3 and Sun-4 systems this path is `/usr/etc/newkey`.

Decryption messages — If your public key password and login password are different, you will see decryption error messages when logging in on Sun386i SunOS 4.0.2 systems. To prevent display of these messages, make your public key password the same as your login password; this is required for your home directory to be mounted properly if it is NFS mounted using the `secure` option and Secure NFS.

Public key errors while running SNAP— If you can log in as yourself, but you receive public key errors while adding a user in SNAP, the problem generally is with the system to which you're adding the user account. Since you logged in successfully, your user public key data is correct. To avoid display of these error messages, delete the public key entry for root of the system you're trying to reach (to do so, edit the `/etc/publickey` file on the YP master, and remake YP). Then, in single-user mode on that same system, delete the `/etc/.rootkey` and `/etc/keystore` files and reboot.



newkey: Command not found — On Sun-3 and Sun-4 systems, the `newkey` command is stored in `/usr/etc`, which is in the superuser's default path. On Sun386i systems, the `newkey` command is stored in `/usr/etc/yp`, which isn't in the default path. To run `newkey` on Sun386i systems, use the full path name to the command.

- ✓ **Problems logging in** — `logintool` used to prevent users from logging in if the login password and public key password involved in user authentication were not identical, or if a user's password was longer than eight characters. This problem is fixed in Sun386i SunOS 4.0.2.

Workaround for 4.0.1 — Users can make their public key passwords the same as their login passwords with the `chkey(1)` command, being sure to use a login password that is no longer than 8 characters. They then must log out and back in again to reregister the new public key.

Keys established by SNAP — As of Sun386i SunOS 4.0.2, user accounts created with SNAP do not have public key or private key information automatically set up for them. SNAP does, however, correctly maintain accounts that use public keys.

Public Key Manipulation and Storage

Since YP is available only in multiuser mode, you must perform most key operations in multiuser mode. This includes changing user or root passwords, since those usually involve re-encrypting the private key in the public key database. The only exceptions to this are deleting the `/etc/keystore` and `/etc/.rootkey` files, which you should do in single-user mode.

Creating a Public Key Entry

Users can establish or change their own public key information by invoking the `chkey(1)` command, and logging out and back in again. (However, if the passwords are different, the user will see decryption warning messages when logging in.)

As a network administrator, you can assign public key information for users and root with the `/usr/etc/yp/newkey` command on the master YP server. For root, you also must issue the `/usr/etc/keyserv` command with the `-n` option to create the `/etc/.rootkey` file. See the on-line man pages for details on these commands.

Never add or delete individual `/etc/.rootkey` or `/etc/keystore` entries; delete only the files, if necessary. You can delete individual `/etc/publickey` entries, but do not add entries to this file except to re-enter the default `nobody` key if it has been deleted. See "nobody Default Key" (page 126) for instructions.

Deleting a Public Key Entry

To delete the public key database entry for a user or root, delete the applicable entry in the `/etc/publickey` file on the YP master and rebuild YP. If you delete a root entry, you must also delete the `/etc/.rootkey` and `/etc/keystore` files on that system, in single-user mode. When you delete a public key entry for a user or root, the default `nobody` key entry is then used for Secure RPC requests.

- ✿ **keystore(5), .rootkey(5)** — There are no on-line or hard-copy man pages describing the `/etc/keystore` and the `/etc/.rootkey` files.

Reference: Security Features Guide (Chapter 6)

On-line Sun386i SunOS 4.0.2 man pages (`man_pages` optional cluster must be loaded) — `chkey(1)`, `newkey(8)`, `keylogin(1)`, `keyserv(8C)`, `publickey(5)`

9.5 Network Time Synchronization

Each Sun workstation uses a clock to keep track of the time and date. These clocks continue to run even when the power is off. Sun386i systems on Sun386i networks try to keep their clocks in close synchronization, using the following rules:

- ◆ If configuration probing is enabled on a client system, that system sets its clock from its boot server each time it boots, via the `/sbin/netconfig` command.
- ◆ Boot servers try to set their clocks from the `timehost` (the master server, by default) when they boot. If boot servers cannot do this, they display a message suggesting that you check the clock time.
- ◆ Time on the `timehost` is maintained by `timehost`'s system clock, and is adjusted manually when necessary.

If your network clocks do not show the correct time, first check that the `timehost` clock is correct by issuing the following command:

```
system: SUPERUSER:1} rsh timehost date
```

To resynchronize any system's clock if the time is only slightly wrong, use the `rdate(8C)` command. If the discrepancy between the time shown and the actual time is more than a few minutes, and your system has configuration probing enabled, you can simply reboot the system to reset the time.

Any system on the network can be designated the `timehost`. Just change the YP hosts (or the system's `/etc/hosts`) definition for `timehost` to the system you want to use. By default on Sun386i networks the `timehost` is the master server.

It's important to keep all system clocks closely synchronized, because NFS relies on time stamps to indicate when it must get updated file information from the server. Other network applications, as well as users, rely on system-supplied time and date information.

On non-Sun386i systems, users must use `rdate` manually to keep systems up to date. If you are using your Sun386i system on a Sun-3/4 network, then you should ensure that the clock is set correctly in the same way, because your Sun386i system most likely does not have configuration probing enabled. (Configuration probing requires a Sun386i YP server. See "Sun386i Probing" in Chapter 5 for details.)

Additional Steps for Reconciling Time Differences on a Master or Standalone

If the date and time on a standalone system or the master server for the network differs from the actual date and time by more than a few minutes, enter the following additional commands after resetting the time using the `date` command (or `rdate`, if the `timehost` is not the master):

```
system: SUPERUSER:1} cd /var/yp
system: SUPERUSER:2} rm *.time *.push
system: SUPERUSER:3} make
system: SUPERUSER:4} exit
```

If you set the date or time on the master server, reboot the boot servers and then reboot all the other systems on the network for this change to take effect throughout the network.



Notes

9.6 Inside SNAP, ASI, and New User Accounts

Automated systems that aspire to ease of use must by their nature hide much complexity. So it is with Automatic System Installation, New User Accounts, and SNAP.

Sun386i networks use the same underlying mechanisms and setup files as other Sun networks. However, because SNAP, Automatic System Installation (ASI), and New User Accounts are automated, it's not always obvious exactly what files have been updated when. In addition, Sun386i ease-of-administration features rely on some new features not found on other Sun systems:

- ◆ Additional YP updating support via `ypupdate(3)`, for use by ASI
- ◆ A YP updater, `rpc.yppbatchupd` (described in `yppbatchupd(8)`), for use by SNAP and `logintool(8)`. This daemon updates YP maps, and also performs a locking service to prevent SNAP users from updating the same maps simultaneously.
- ◆ A “home directory agent,” `rpc.user_agentd` (described in `user_agentd(8)`), for use by SNAP and `logintool(8)`
- ◆ Daemons for allocating various network identifiers such as `uid_allocd(8C)`, `gid_allocd(8C)`, and `ipallocd(8C)`. The `ipallocd(8C)` daemon is used by ASI and SNAP to allocate unique IP addresses to new systems being installed on the network. The `uid_allocd` and `gid_allocd` daemons allocate new user and group IDs, respectively.
- ◆ RARP (Reverse Address Resolution Protocol) daemon support (`rarpd(8)`) for Dynamic RARP (Internet standards are not yet finalized). Automatic System Installation uses the modified RARP daemon and DRARP to assign addresses to systems trying to install themselves, and to report errors encountered while assigning those addresses.
- ◆ `pnpd(8C)` for automatic system installation, which includes the “diskless agent” for use in error recovery (`client(8C)` and SNAP also use this agent). The `pnpd(8C)` daemon looks up configuration information for systems at boot time, and also communicates requests to the master server when a new system is installed.
- ◆ `tftpd(8C)`, modified for Automatic System Installation of diskless systems
- ◆ Miscellaneous YP improvements such as `yppsync`, which collects the most current YP maps

This section explains the procedures used by SNAP, Automatic System Installation, and New User Accounts. Chapter 10 outlines the contents of various files associated with Sun386i networking.

Diagnosing Errors

In the event that something goes wrong, Sun386i ease-of-administration tools generally produce appropriate error messages:

SNAP errors – For SNAP, read the diagnostic messages in the alerts, and choose the **Why?** button for more detail. In some cases the **Why?** button expands to several levels of detail.

logintool and New User Accounts errors – For errors that occur during account creation from `logintool(8)`, see the file `/var/adm/messages` on the local system if it has a disk, or on the server if the system is diskless.

Errors at bootup – Errors that occur at boot time and during Automatic System Installation are listed in the `/var/adm/messages` file. See the Sun386i Field Service Manual for explanations of error codes that appear in this file and on the screen.

If you want to see traditional SunOS messages when booting, you can turn on the system's verbose message mode. See Chapter 4 for details.

For help diagnosing errors, see the applicable sections of Sun386i SNAP Administration.



Adding a Network Client Through SNAP

The process of adding a network client through SNAP happens in two stages.

In Part A, SNAP updates the appropriate YP maps with information about the new system. In Part B, shown on the following pages, the new system sets up its own local configuration files based on the information that SNAP has placed in the YP database.

Part A – What SNAP Does

SNAP performs the following procedures when you add a new network client. The headings below match those from the diagram.

Read YP Maps – As soon as you select the **Systems** category, SNAP reads the YP maps for the various files listed in the diagram. Files in parentheses () are read but will not be rewritten.

Allocate IP Address – You can assign your own IP address for a system by specifying the address in SNAP. If you leave the system number address field in SNAP blank, new IP addresses are assigned automatically by a daemon (`ipallocald`). To prevent possible duplicates, this daemon also matches the generated address against existing addresses listed in the `hosts` and `ipalloc.netrange` maps, and against an internal cache of all current IP address client requests on the network.

Update YP Maps – The listed files are completely rewritten, with additional entries now included for the new system. The same daemon that updates these files (`rpc.ybatchupd`) then remakes YP, distributing the new maps to all slave servers.

Part A Complete – At this point, the network is prepared to recognize the system when it is added to the network. SNAP rereads the YP maps and is ready for the next operation. You can quit SNAP or leave it running—all the requisite YP files for the new system are now up to date.

What Happens When You Add a Network Client via SNAP

What You Do Part A – What SNAP Does

- Run SNAP**
- 1 Start SNAP from any system
 - 2 Select **Systems** category
 - 4 Choose **Add** & enter information
 - 5 Choose **Save**
 - 7 Choose **Confirm**



3 Read YP Maps

Read maps for:
 systems
 hosts
 ethers
 (bootservers)
 (bootparams)
 (networks)
 (netmasks)
 (publickey)
 (ypservers)

(These files are read but not rewritten)

(Nothing added or updated yet on the network client)

6 Allocate IP Address

ipallocald daemon allocates address

8 Update YP Maps

Update /etc/systems

Update /etc/hosts

Update /etc/ethers

Rebuild YP maps

9 Part A Complete

Continued on the following pages...



Part B – What Happens When a User Starts the System

Before it is turned on, all a new system “knows” about is its Ethernet address. The Ethernet address for each system is recorded in the hardware, and is guaranteed unique for each system. The remainder of the information required to install the system is obtained from the network.

Once you have prepared the network for the new system, the new system performs the following tasks when someone turns it on for the first time.

Connected to Network? – The system probes to see if it is connected to the network.

Get IP Address – The program `/sbin/netconfig`, started from `/etc/rc.boot`, broadcasts a Dynamic RARP (DRARP) request for the IP address of this system. This program determines that the client has not yet been installed by looking at a special flag (`MUST_SETUP`) in `/etc/net.conf`.

Look up Client – The first server (master or slave) to answer the request becomes the temporary boot server that will provide the IP address to the new client.

The `rarpd` daemon running on the boot server uses the client’s Ethernet address as a key to look up the system’s assigned IP address. The IP address is then returned to the client.

How the lookup works: The daemon first matches the Ethernet address with an entry in the YP `ethers.byaddr` map to determine the client’s host name. (For this reason, the `ethers` maps on Sun386i networks contain the Ethernet address of every machine on the network.) The daemon then uses this host name to look up the client’s IP address in the YP `hosts.byname` map.

For additional notes on the `ethers` and `hosts` maps, see Chapter 10.

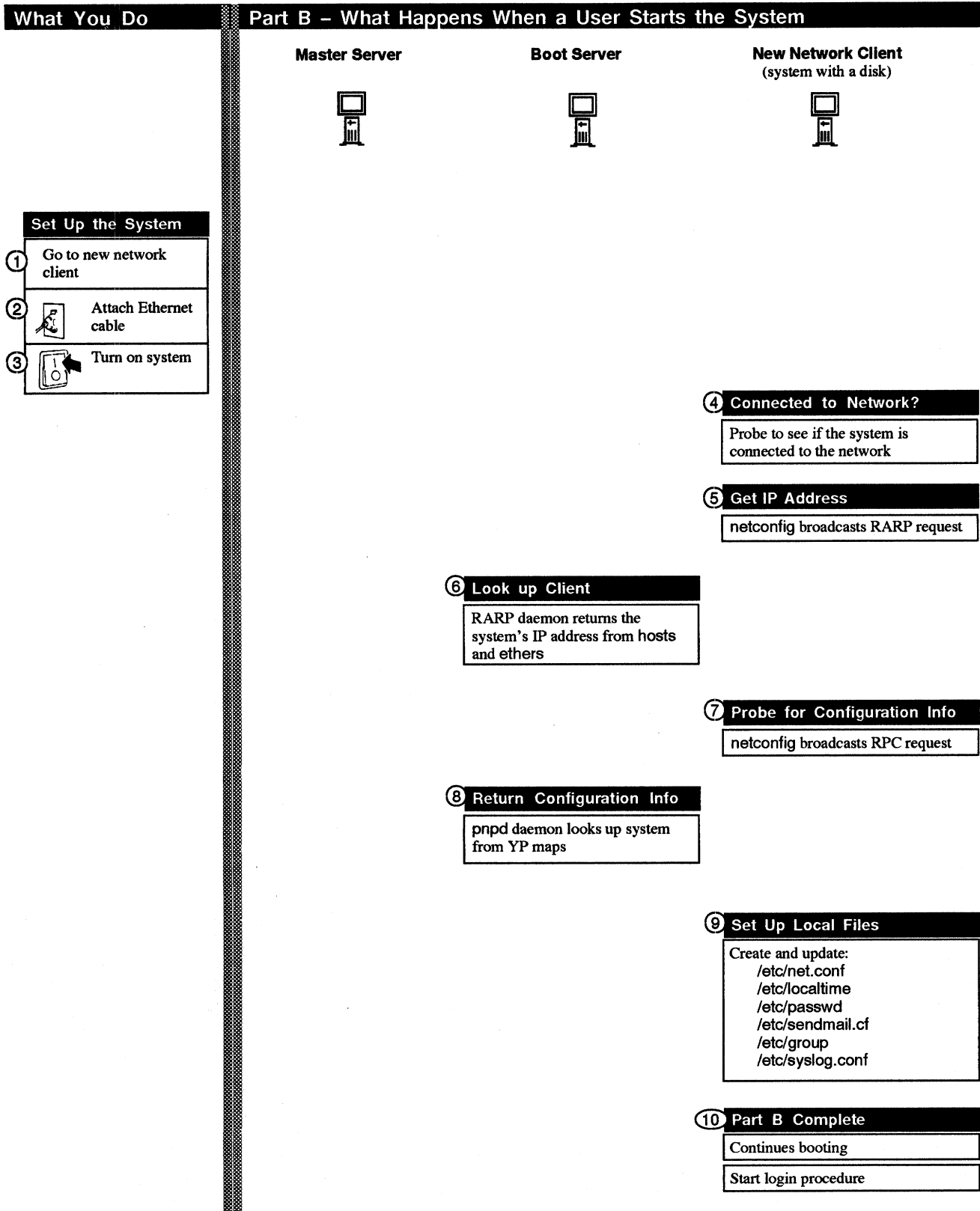
Probe for Configuration Information – Once the `netconfig` program knows the IP address, it sends out a request for additional configuration information such as the name of the domain, and the host name. Again, the first server to answer becomes the temporary boot server that manages the configuration information.

Return Configuration Information – The `pnpd` daemon on the boot server looks up configuration information, and passes it back to `netconfig` on the client system.

Set Up Local Files – Based on the information gathered during configuration probing, the client creates and updates its `/etc/net.conf` file, and then creates the additional files shown in the diagram. As part of the procedure, the special `MUST_SETUP` flag is removed from `/etc/net.conf`.

Part B Complete – The system is now fully installed. It finishes booting and then displays the login prompt.

What Happens When You Add a Network Client via SNAP (cont'd)



Installing a Network Client via ASI

The process of adding a new network client involves three systems:

- ◆ The master server, where maps are updated and the name and IP address of the client is allocated
- ◆ The boot server, from which the new system gets some initial configuration information. The boot servers in steps 6, 8, 9a, and 10 can actually be different systems, but their role in each of these steps is to provide setup information for the new client.
- ◆ The new network client, which issues the configuration requests.

What ASI Does

Automatic System Installation performs the following procedures when you add a new network client. This diagram assumes that the domain `ip_address_allocation` policy is set to `drarp_only`, and the `pnp` policy is set to `unrestricted` (the defaults on Sun386i networks).

To begin with, all a new system “knows” about is its Ethernet address. The Ethernet address for each system is recorded in the hardware, and is unique for each system. The remainder of the information required to install the system is obtained from the network.

The headings below match those from the diagram.

Connected to Network? – The system probes to see if it is connected to the network.

Request IP Address – The program `/sbin/netconfig`, started from `/etc/rc.boot`, broadcasts a Dynamic RARP (DRARP) request for the IP address of this system. This program determines that the client has not yet been installed by looking at a special flag (`MUST_SETUP`) in `/etc/net.conf`.

Recognize Client as New – The first server (master or slave) to answer the request becomes the temporary boot server that will manage IP address allocation to the new client.

The RARP daemon running on the boot server uses the client’s Ethernet address to attempt a lookup of the system’s IP address from the `ethers` and `hosts` maps. Not finding an entry for the system (since it hasn’t yet been installed), the server recognizes the client as new and requests a new IP address from the master server (see next heading). The RARP daemon then broadcasts this IP address to the network to double-check that no other systems are using it. (Note that this broadcast validation is only done when adding a system through ASI—this double-checking is not performed by SNAP.)

Allocate IP Address – New IP addresses are assigned automatically by a daemon (`ipallocald`) that generates an address, and then matches it against those listed in the `hosts` and `ipalloc.netrange` maps as well as against an internal cache kept by the `ipallocald` daemon.

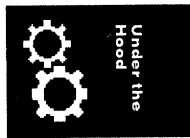
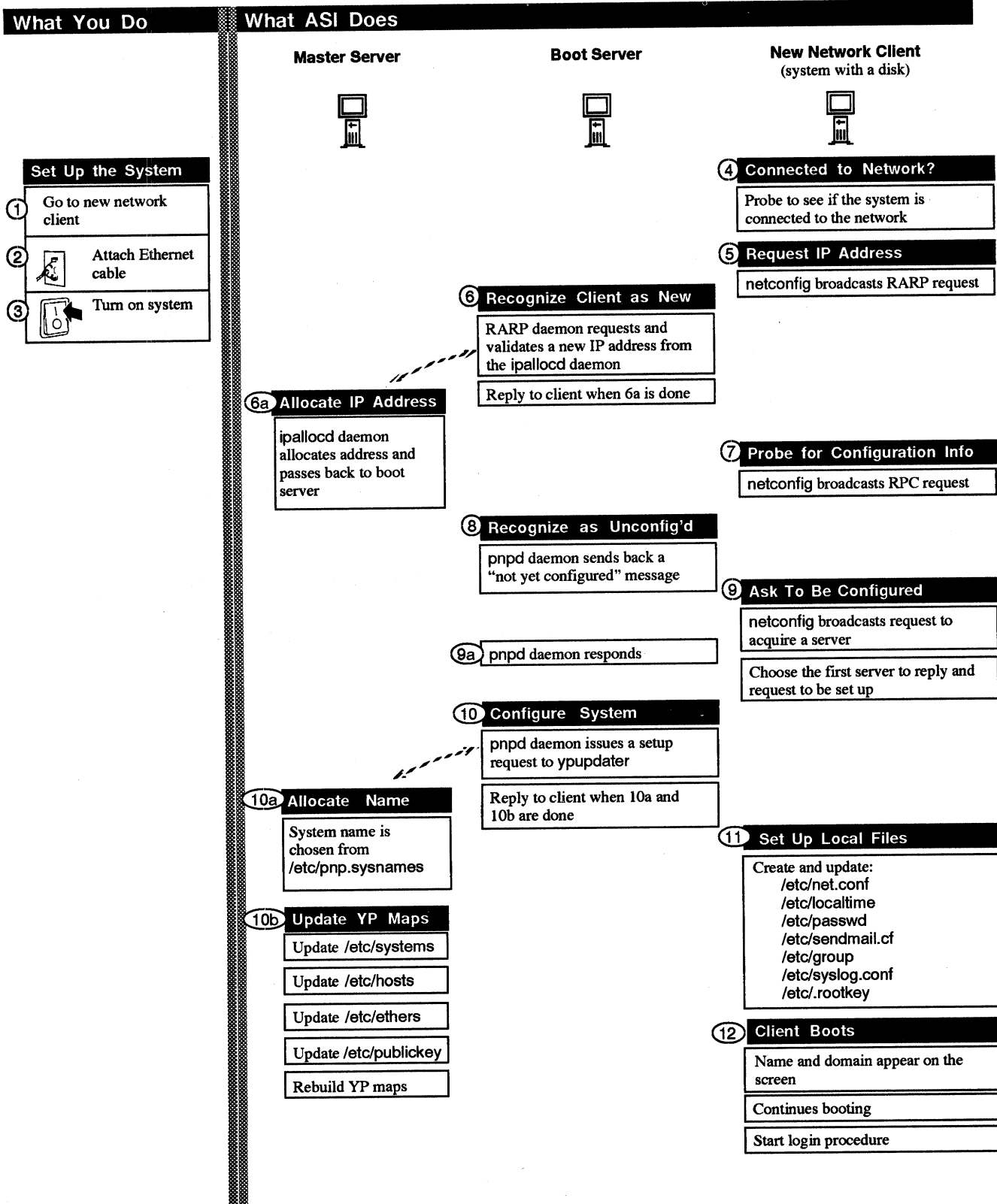
Probe for Configuration Information – Once the `netconfig` program knows the IP address, it broadcasts an RPC request for additional configuration information such as the name of the domain, the host name, and the time zone.

Recognize as Unconfigured – Again, the first server (master or slave) to answer the request becomes a temporary boot server that will provide initial configuration information to the new client.

The `pnpd` daemon running on the boot server uses the client’s Ethernet address as a key to look up the system’s assigned IP address. The daemon first tries to match the Ethernet address with an entry in the YP `ethers.byaddr` map to determine the client’s host name. Because this is a new system, an entry will not exist for it, and the daemon will inform the new client that it is not yet set up in the YP maps.

(continued)

What Happens When You Add a Network Client via ASI



Installing a Network Client via ASI (cont'd)

Ask to Be Configured – The client broadcasts an RPC request to acquire a server, and accepts the first response from the pnpd daemon on a willing boot server. The client then issues an RPC request to be set up by the server.

Configure System – The pnpd daemon on the boot server issues a request to the master server to set up the required maps, and then returns the domain name, host name, time zone, and network role to the client.

Allocate Name – Names are assigned from the `/etc/pnp.sysnames` file. See `pnp.sysnames(5)` for details.

Update YP Maps – Entries for the new system are added to the listed files on the master server. The same daemon that updates these files (`rpc.yppupdated`) then remakes YP, distributing the new maps to all slave servers.

Set Up Local Files – Based on the information gathered from the configuration request, the client creates and updates its `/etc/net.conf` file, and then creates the additional files shown in the diagram.

Client Boots – The system is now fully installed. It displays its name and domain, finishes booting, and then displays the login prompt.

ASI Without a Network

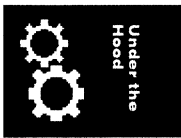
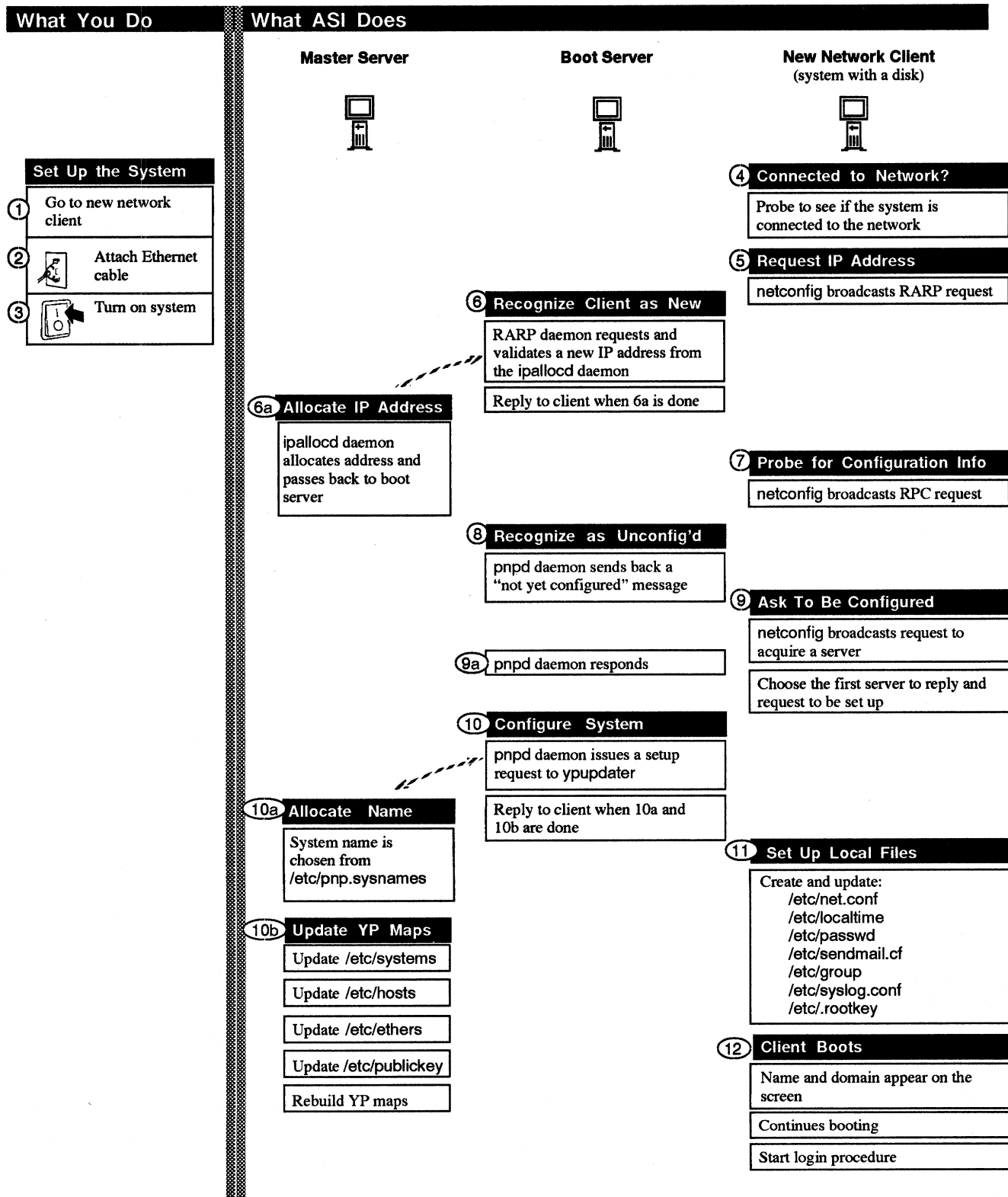
On a standalone system the procedure is the same, except that all of the master server and boot server operations take place on the new system. A Sun386i standalone system is effectively a “network of one system.”

ASI for Setting Up a Master Server

As on a standalone system, the procedure for setting up a master server is basically the same as for setting up a network client.

Exceptions: The contents of some files are different, and all operations in the diagram take place on the system you’re setting up as the master server.

What Happens When You Add a Network Client via ASI
(This diagram is identical to the one presented on page 137.)



Adding a Diskless Client Through SNAP

The process of adding a diskless client through SNAP really happens in two stages.

In Part A, as shown in the diagram, SNAP updates the appropriate YP maps with information about the new system. In Part B, shown on the following pages, the new system sets up its own local configuration files based on the information that SNAP has placed in the YP database.

See Sun386i SNAP Administration (June 1989 Edition) or the on-line Sun386i Help Viewer for further information on how to add diskless clients via SNAP.

Assumptions When Running SNAP

Before adding a diskless client through SNAP, you must have already enabled the boot server via SNAP to accept an additional diskless client.

(If the intended boot server isn't enabled, or isn't set up to accept more than the diskless clients it currently serves, its name will be dimmed in the **Bootserver** menu and you won't be able to add the diskless client to it.

Additionally, if no boot server can accept any more diskless clients, the **Diskless Client** entry in the **Network Role** menu will be dimmed, and you won't be able to select it.

Use the slide bar in a server's SNAP profile to indicate how many diskless clients that server will accept.)

Part A – What SNAP Does

SNAP performs the following procedures when you add a new diskless client. The headings below match those from the diagram.

Read YP Maps – As soon as you select the **Systems** category, SNAP reads the YP maps for the various files listed in the diagram. Files in parentheses () are read but will not be rewritten.

- ✓ Note that SNAP can read files from any YP server, including slaves. As of 4.0.2 it no longer tries to bind to the YP master.

Allocate IP Address – You can assign your own IP address for a system by specifying the address in SNAP. If you leave the system number field in SNAP blank, a new IP address is assigned automatically by a daemon (`ipallocald`). To prevent possible duplicates, this daemon also matches the generated address against existing addresses listed in the `hosts` and `ipallocal.netrange` maps, and against an internal cache kept by the `ipallocal` daemon.

Set Up System Directories – The `pnpd` daemon (`rpc.pnpd`) sets up system files for the diskless client in various subdirectories under `/export` on the boot server. These directories are added to the diskless client's `/etc/exports` file and exported. Also added to the client's `/etc/exports` and `/etc/fstab` files are `/usr`, `/usr/cluster` (if applicable), and `/usr/local`.

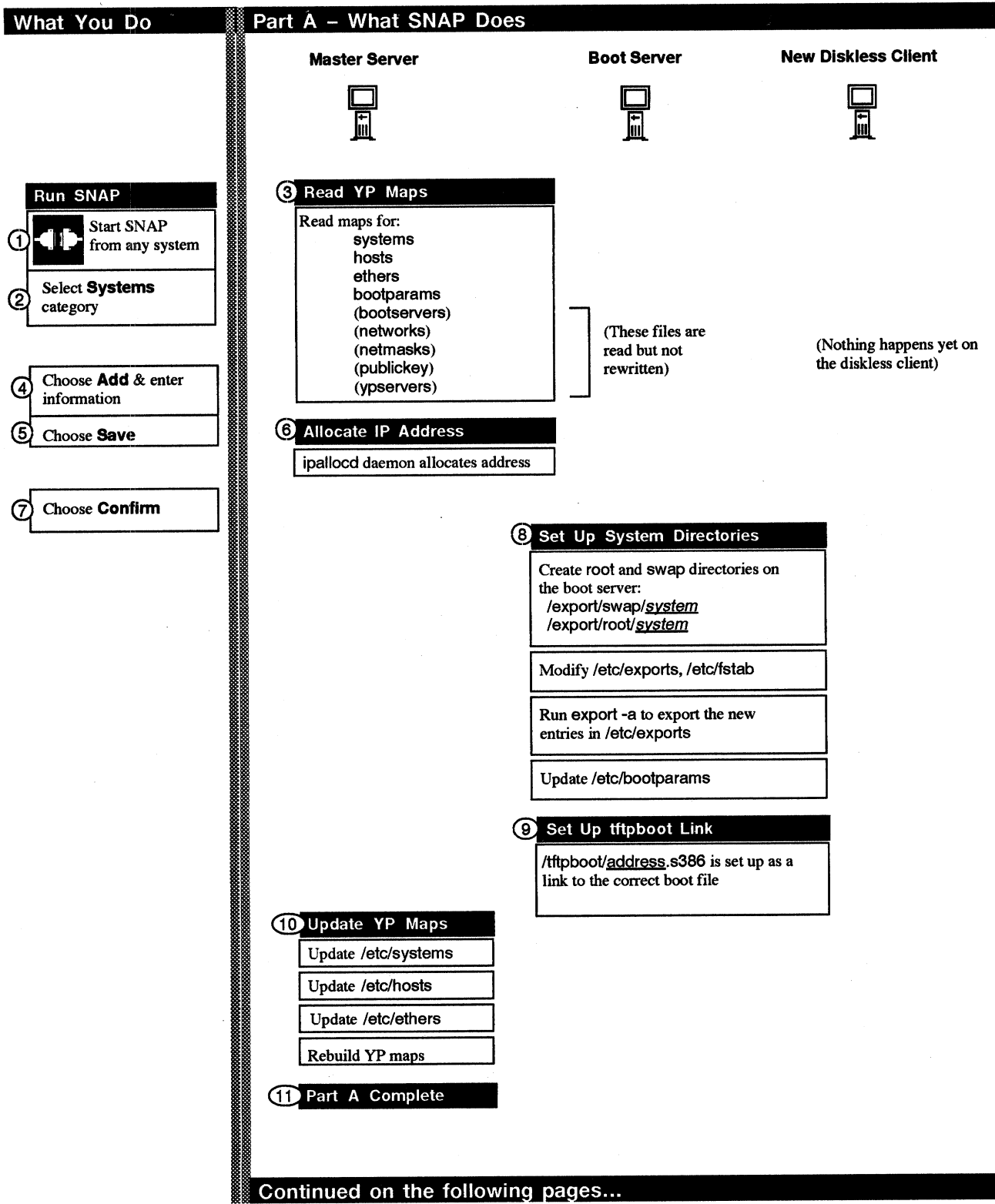
(Note that the entries in `/export` are really symbolic links to directories in other locations, as determined by entries in `/etc/where`. See page 116 for details.)

Set Up `tftpboot` Link – The `pnpd` daemon sets up the link to the appropriate boot file for this SunOS version and architecture.

Update YP Maps – The listed files are completely rewritten, with additional entries now included for the new system. The same daemon that updates these files (`rpc.yppatchupd`) then remakes YP, distributing the new maps to all slave servers.

Part A Complete – SNAP rereads YP maps and is ready for the next operation. At this point, the network is prepared to recognize the system when it is added to the network. You can quit SNAP or leave it running. All the requisite YP files for the new system are now up to date.

What Happens When You Add a Diskless Client via SNAP



Continued on the following pages...



Part B – What Happens When a User Starts the System

Once you have prepared the network for the new system, the new system performs the following tasks when someone turns it on for the first time. The headings below match those from the diagram.

Connected to Network? – The system probes to see if it is connected to the network.

Request IP Address – The PROM broadcasts a Dynamic RARP (DRARP) request for the IP address of this system.

Look up Diskless Client – The first server (master or slave) to answer the request becomes the boot server for the diskless client.

The `rarpd` daemon running on the boot server uses the client's Ethernet address as a key to look up the system's assigned IP address. The IP address is then returned to the diskless client.

How the lookup works: The daemon first matches the Ethernet address with an entry in the YP `ethers.byaddr` map to determine the client's host name. The daemon then uses this host name to look up the client's IP address in the YP `hosts.byname` map.

Standard Boot Sequence – The PROM initiates the standard diskless boot sequence.

Standard Boot Interaction – The boot server performs normal boot steps. See `boot(8)` for details.

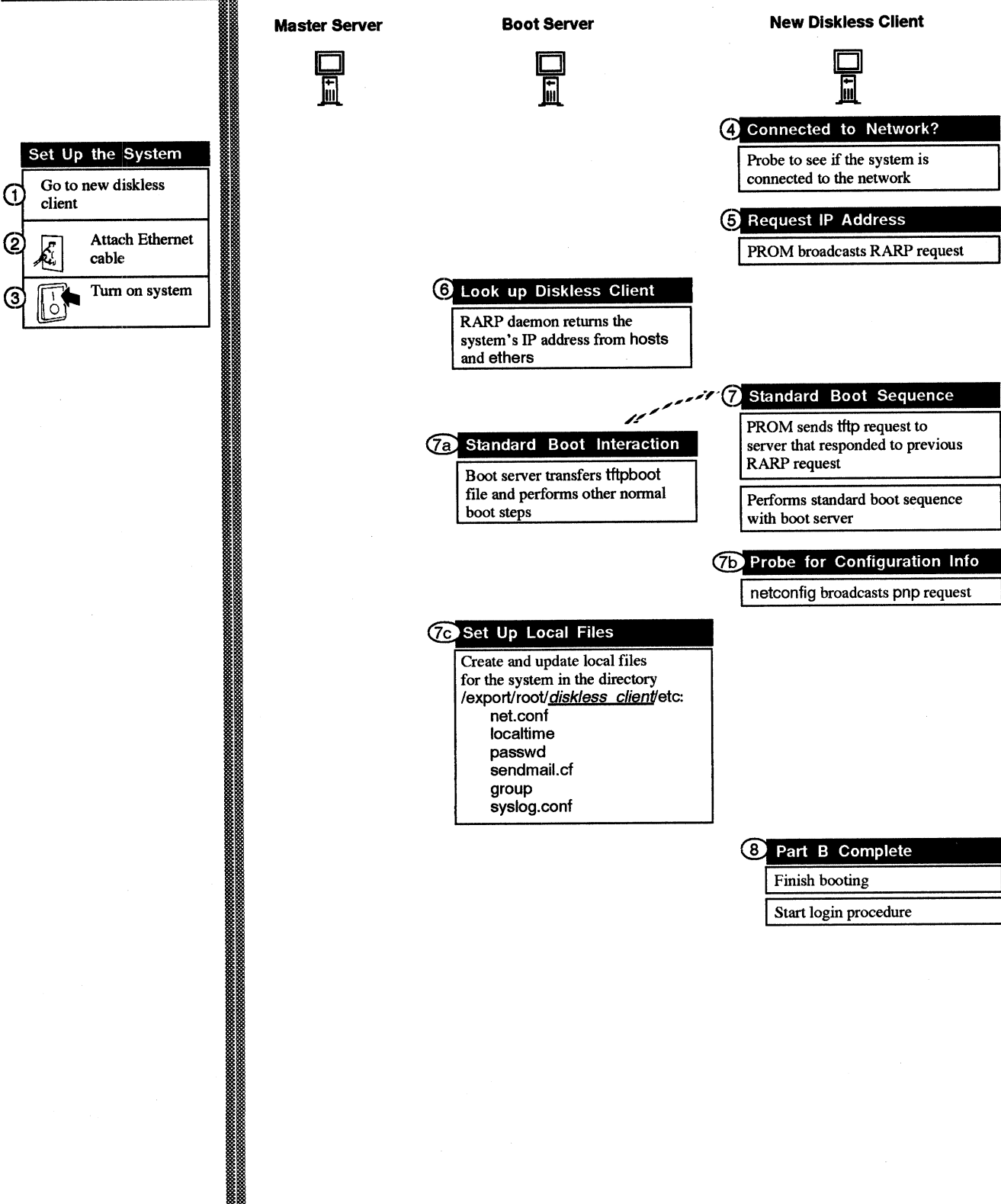
Probe for Configuration Information – Once booting has begun, `rc.local` starts `/sbin/netconfig`, which broadcasts an RPC request for additional configuration information such as the name of the domain, the host name, and the time zone.

Set Up Local Files – Local files for a diskless client are stored on the boot server in `/export/root/diskless_client/etc`. Based on the information gathered during configuration probing, the client creates and updates its `net.conf` file, creates the `passwd` file and writes the root password (the encoded `hostid`) into it. The client then creates the additional files shown in the diagram.

Part B Complete – The system is now fully installed. It finishes booting and then displays the login prompt.

What Happens When You Add a Diskless Client via SNAP

What You Do Part B – What Happens When a User Starts the System



Adding a Diskless Client Through ASI

The process of adding a diskless client involves three systems:

- ◆ The master server, where maps are updated and the name and IP address of the client is allocated
- ◆ The boot server, from which the diskless client boots and where all its local files are stored
- ◆ The diskless client, which issues the initial boot and configuration requests

What ASI Does

Automatic System Installation performs the following procedures when you add a new diskless client. The headings below match those from the diagram.

Connected to Network? – The system probes to see if it is connected to the network.

Request IP Address – The PROM broadcasts a RARP request, which will receive no response because the diskless client is new and thus is unknown on the network. The PROM then broadcasts a Dynamic RARP (DRARP) request to get its IP address of this system.

Recognize Client as New – The RARP daemon running on the boot server recognizes the client as new and so requests a new IP address from the master server (see next step). The RARP daemon then broadcasts this IP address to the network to double-check that no other systems are using it. (Note that this broadcast validation is only done when adding a system through ASI—this double checking is not performed by SNAP.)

Allocate IP Address – New network addresses are assigned automatically by a daemon (`ipallocald`) that generates an address and then matches it against those listed in the `hosts` and `ipalloc.netrange` maps. The `ipallocald` daemon also matches generated addresses against its own internal cache of current IP address client requests on the network.

Request Boot File Download – This is the standard request that a diskless client issues whenever it is booted. The PROM sends a `tftp` request to boot from the file `ip_address.s386` on the boot server.

Respond with Boot File – Because the file `ip_address.s386` does not exist, the boot server will respond with a special boot file, `/tftpboot/pnp.s386`.

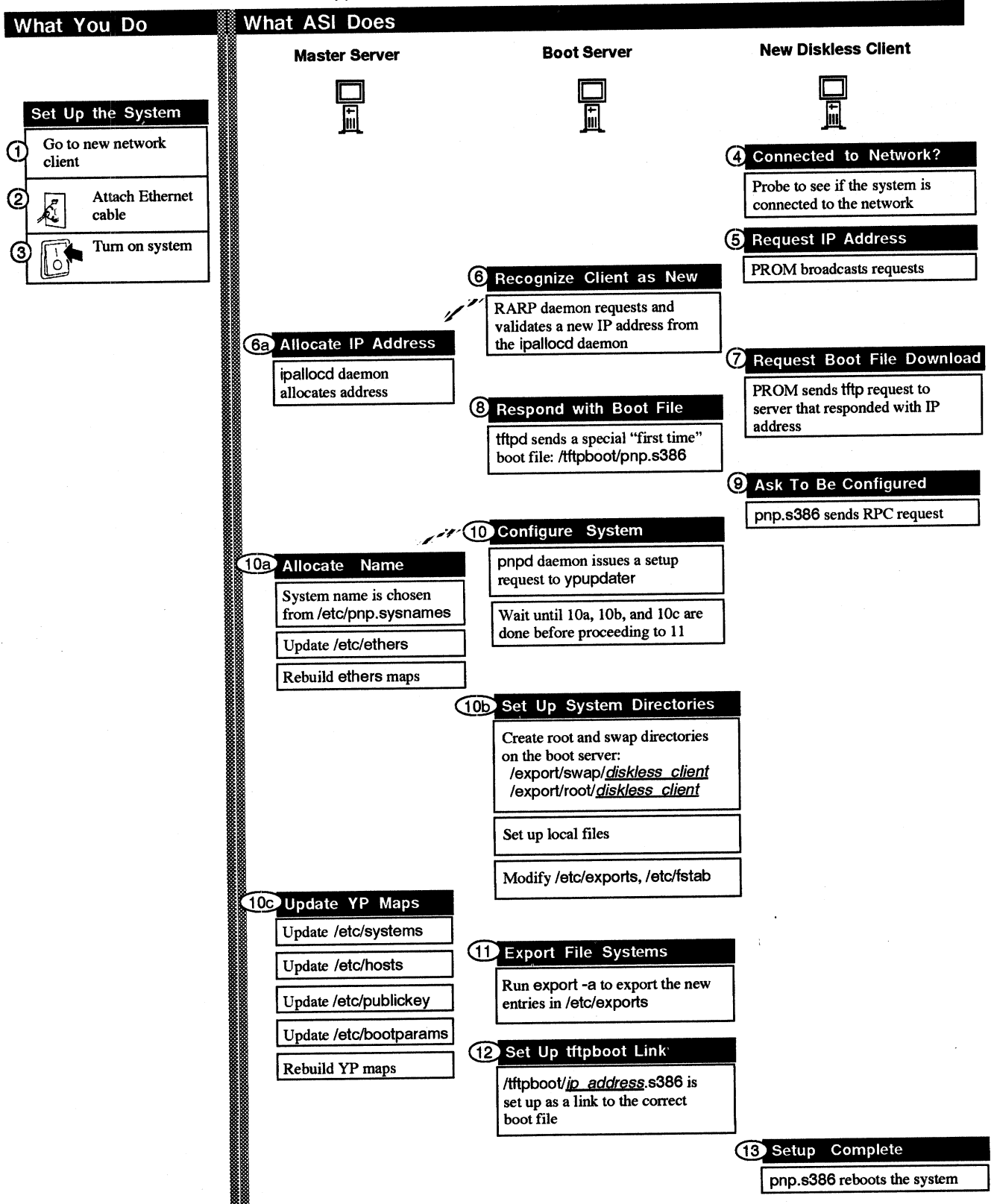
Ask To Be Configured – The program `pnp.s386` sends an RPC request for the system to be configured.

Configure System – The `pnpd` daemon on the boot server issues the setup request on behalf of the new diskless client.

Allocate Name – The system name is chosen from a pool of available names in `pnp.sysnames(5)`. Next, the Ethernet address and name are added to the `ethers` map.

(continued)

What Happens When You Add a Diskless Client via ASI



Adding a Diskless Client Through ASI (cont'd)

Set Up System Directories – System files for a diskless client are stored in various subdirectories under `/export` on the boot server. These directories are added to the diskless client's `/etc/exports` file. Also added to the client's `/etc/exports` and `/etc/fstab` files are `/usr`, `/usr/cluster` (if applicable), and `/usr/local`.

Note that the entries in `/export` are symbolic links to directories in other locations, as determined by entries in `/etc/where`. See page 116 for details.

As part of these steps, the standard files set up in `/export/root/system/etc` are:

- ◆ `net.conf`
- ◆ `localtime`
- ◆ `passwd`
- ◆ `group`
- ◆ `syslog.conf`
- ◆ `sendmail.cf`

The `net.conf` file is updated with information gathered during configuration, and the root password (the encrypted `hostid`) is written to the `passwd` file at this time.

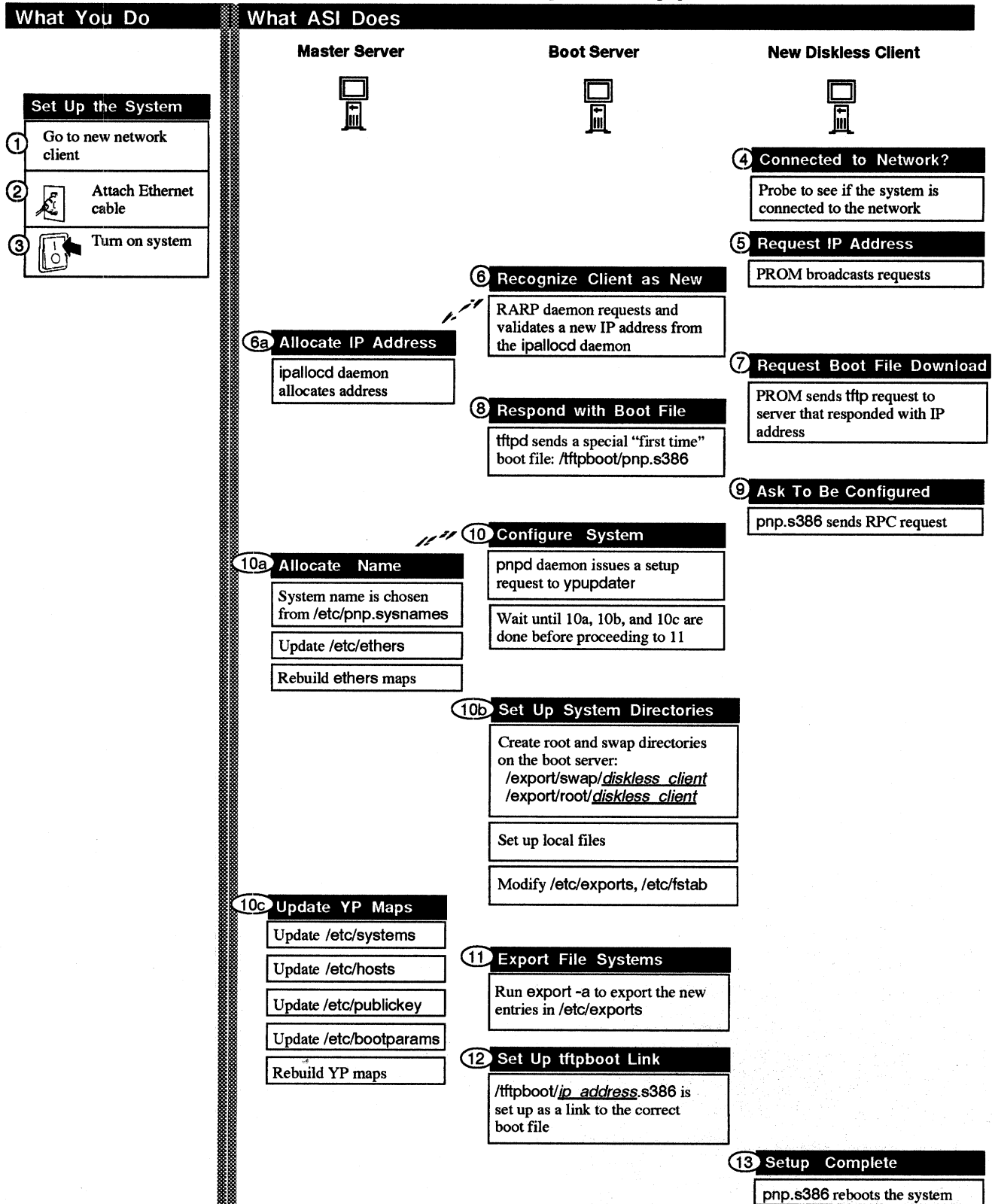
Update YP Maps – Entries for the new system are added to the listed files. The same daemon that updates these files (`rpc.yupdated`) then remakes YP, distributing the new maps to all slave servers.

Export File Systems – The new directories set up in `/etc/exports` are exported so as to be available to the diskless client.

Set Up `tftpboot` Link – Sets up the link to the appropriate boot file for this SunOS version and architecture.

Setup Complete – The system is now fully installed. The program `pnps386` reboots the system now that the proper files are set up on the server to allow a normal diskless boot.

What Happens When You Add a Diskless Client via ASI
(This diagram is identical to the one presented on page 145.)



Adding a User through SNAP

The process of adding a user involves three systems:

- ◆ The master server, from which maps are read and updated
- ◆ The group's home directory server, which stores default files that will be copied to the new user's home directory
- ◆ The new user's home directory server, where the user's home directory will be set up

What SNAP Does

SNAP performs the following procedures when you create a new user account. The headings below match those from the diagram.

Read YP Maps – As soon as you select the **Users** category, SNAP reads the YP maps for the various files listed in the diagram. Files in parentheses () are read but will not be rewritten.

- ✓ Note that SNAP can read files from any YP server (including slave servers). As of 4.0.2, it no longer tries to bind to the YP master.

Allocate UID – New UIDs are assigned automatically by a daemon (`uid_allocd`) that generates a user ID and then matches it against those listed in the `passwd.byuid` map. The `uid_allocd` daemon also matches this generated UID against its own internal cache of other current UID allocations on the network.

Set Up Group Directories – Although users know their home directories as `/home/username`, home directories are physically organized by user group. For instance, the home directory for `mtravis` in the group `users`, is stored on `mtravis`' home directory server as `/files/home/users/mtravis`.

The user agent daemon (`rpc.user_agentd`) creates the appropriate group directory if it does not yet exist on the user's home directory server. Note that this group directory is frequently empty (except for containing home directories), and exists solely for organizational purposes. A group account's true home directory—where group default files are stored—may be on an entirely different system.

Set Up Home Directory – The home directory is set up on the workstation you designate as the "home directory server" for this user. The user agent daemon creates the user's home directory and then sets up a symbolic link so that it can be exported via `/export`. See "Inside the Sun386i File System" at the beginning of this chapter, for more information about exporting conventions.

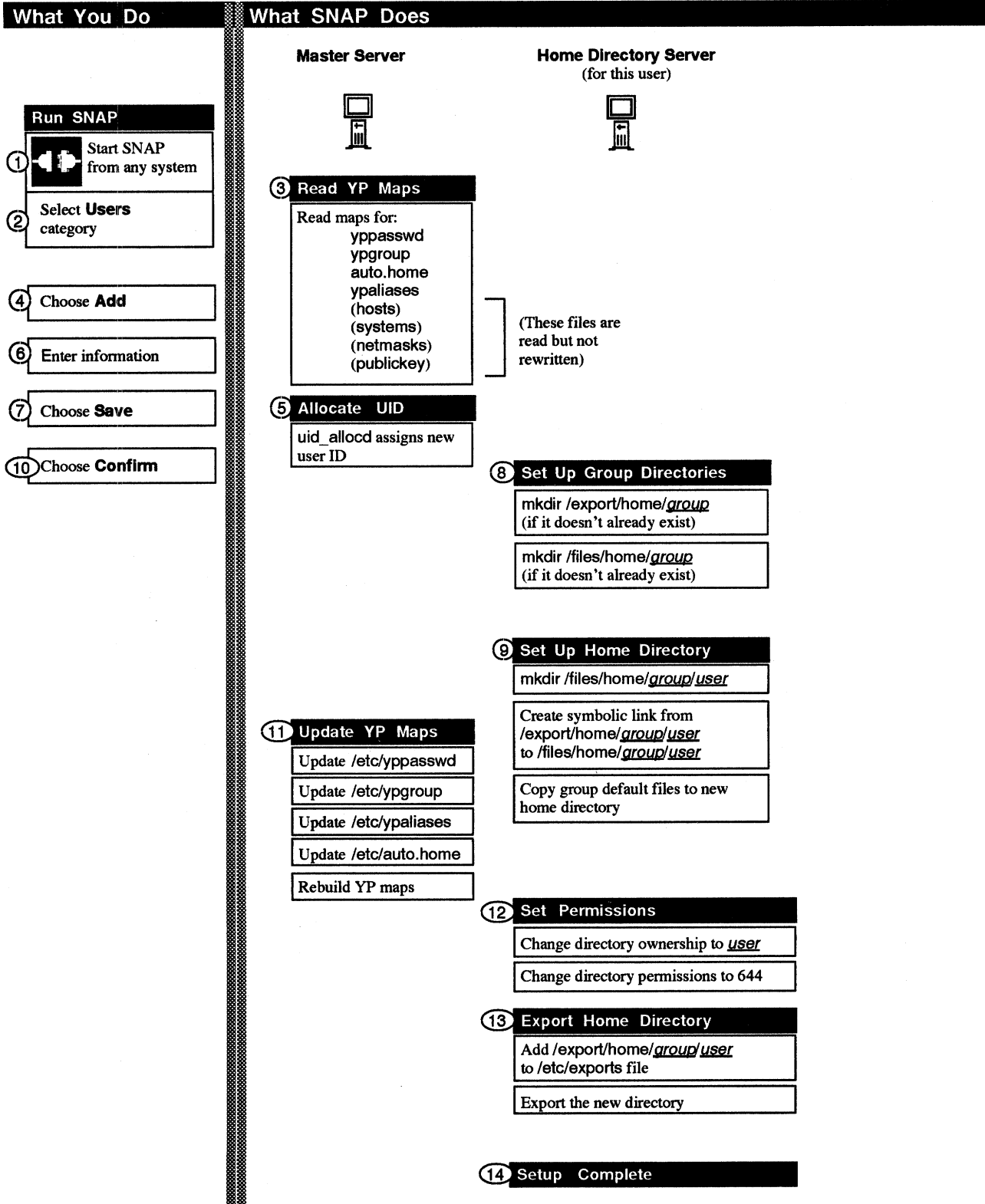
Also in this step, default files and directories for a new user are copied from the directory `/home/group/defaults` (`/files/home/group/group/defaults` on the group's home directory server). You can set up custom environments for new users by editing these files or adding new ones. (If you want the system to perform additional setup tasks during new account creation, add instructions to the `copy_home` script in `/home/group`.)

Update YP Maps – These files are completely rewritten, with additional entries now included for the new user. The same daemon that updates these files (`rpc.yppatchupd`) then remakes YP, distributing the new maps to all slave servers.

Set Permissions – The user agent daemon sets appropriate directory permissions for the new user.

(continued)

What Happens When You Add a User via SNAP



Adding a User Through SNAP (cont'd)

Export Home Directory – The user agent daemon exports the home directory so that it will be available to all systems in the domain. Once exported, this directory is accessible on the network as `/home/username` via the automounter.

Setup Complete – SNAP rereads YP maps and is ready for the next operation. As soon as SNAP redisplay its screen, the user account has been set up.

- ✓ **Secure RPC** – In Sun386i SunOS 4.0.2, SNAP no longer sets up entries for users in `/etc/publickey`. Creating and changing passwords in SNAP now works just like the `passwd(1)` command. SNAP still maintains `publickey` information for existing accounts that have public keys.

You can add `publickey` entries for new users with the `chkey(1)` command.

man page for `copy_home` – Sun386i SunOS 4.0 and 4.0.1 do not have a man page for `copy_home`. SunOS 4.0.2 does include a man page on this script.

Additional Notes about SNAP and Users

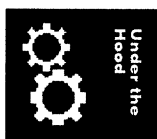
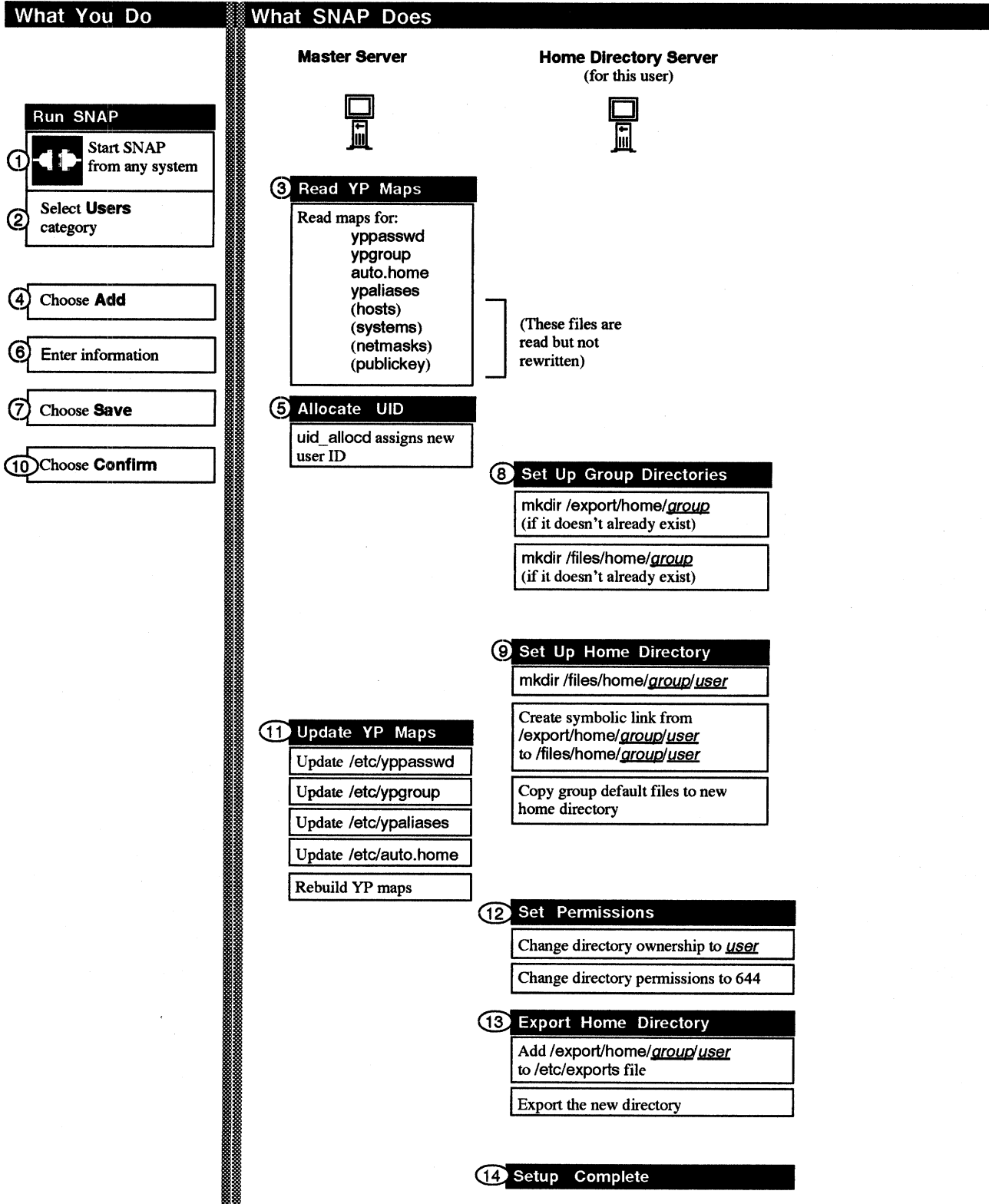
The following procedures use a variation of the steps performed when SNAP adds a user. Because all the same files are involved, separate diagrams and descriptions are not included:

- ◆ **Changing a user's primary group** – YP maps are updated and the user's home directory is moved on the home directory server (from `/files/home/oldgroup/user` to `/files/home/newgroup/user`). The symbolic links for the user in `/export/home` are also changed appropriately and the automounter's `auto.home` map is updated with the new home directory path.
- ✓ **In Sun386i SunOS 4.0.1**, the `auto.home` map was not properly updated if you changed a user's group via SNAP, and users would see the following message upon logging in:

```
No Directory! Logging in with home=/  
See page 36 of the Sun386i SunOS 4.0.1 Owner's Bulletin for a workaround. Sun386i SunOS 4.0.2 supports any combination of simultaneous changes to user name, primary group, or home directory server.
```
- ◆ **Changing a user name** – Home directory is moved from `/files/home/group/oldusername` to `/files/home/group/newusername`.
- ◆ **Changing the home directory server** – Home directory is moved to new system.
- ◆ **Removing a user** – All YP maps set up when the user is added are updated accordingly. User files are removed from the home directory server. Additionally, any `publickey` entry is removed.

The manual counterparts to these procedures are covered in Sun386i Advanced Administration.

What Happens When You Add a User via SNAP
 (This diagram is identical to the one presented on page 149.)



Adding a User via New User Accounts

The process of adding a user through the New User Accounts process involves two systems:

- ◆ The master server, from which maps are read and updated, and where the default group directory (for the group `users`) normally resides
- ◆ The user's machine, which will become his or her home directory server

What New User Accounts Does

The New User Accounts facility performs the following procedures when you create a new user account. The headings below match those from the diagram.

Standard Boot Sequence – The system runs its standard boot sequence, reading `/etc/net.conf`, probing the network for configuration information, and executing the commands in `/etc/rc.boot`, `/etc/rc`, and `/etc/rc.local`.

As part of booting, `init(8)` reads `/etc/ttytab(5)`, which contains a special `getty(8)` line for the console:

```
console    "/usr/etc/getty -n -s -l std.9600" sun on secure
```

The `-n` flag instructs `getty` to bring up `logintool(8)` (and the New User Accounts screens) in lieu of running `login(8)`.

Allocate UID – New UIDs are assigned automatically by a daemon (`uid_allocd`) that generates a user ID and then matches it against those listed in the `passwd.byuid` map. The `uid_allocd` daemon also matches this generated UID against its own internal cache of other current UID allocations on the network.

Set Up Group Directories – The user agent daemon (`rpc.user_agend`) creates the appropriate group directory if it does not yet exist on the system (or on the bootserver, if it's a diskless machine). Note that this group directory is frequently empty (except for containing home directories), and exists solely for organizational purposes. A group account's true home directory—where group default files are stored—may be on an entirely different system.

Set Up Home Directory – Because the user is being added through the New User Accounts feature, the user agent daemon places the home directory on the machine from which the user logs in (or on the boot server for that machine, if the new user logs in to a diskless system).

Also in this step, default files and directories for a new user are copied from the `/home/users/defaults` directory. You can set up custom environments for new users by editing these files or adding new ones.

If you want the system to perform additional setup tasks during new account creation, add instructions to the `copy_home` script in `/home/users`.

Update YP Maps – These files are completely rewritten, with additional entries now included for the new user. The same daemon that updates these files (`rpc.yppatchupd`) then remakes YP, distributing the new maps to all slave servers.

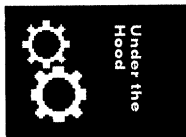
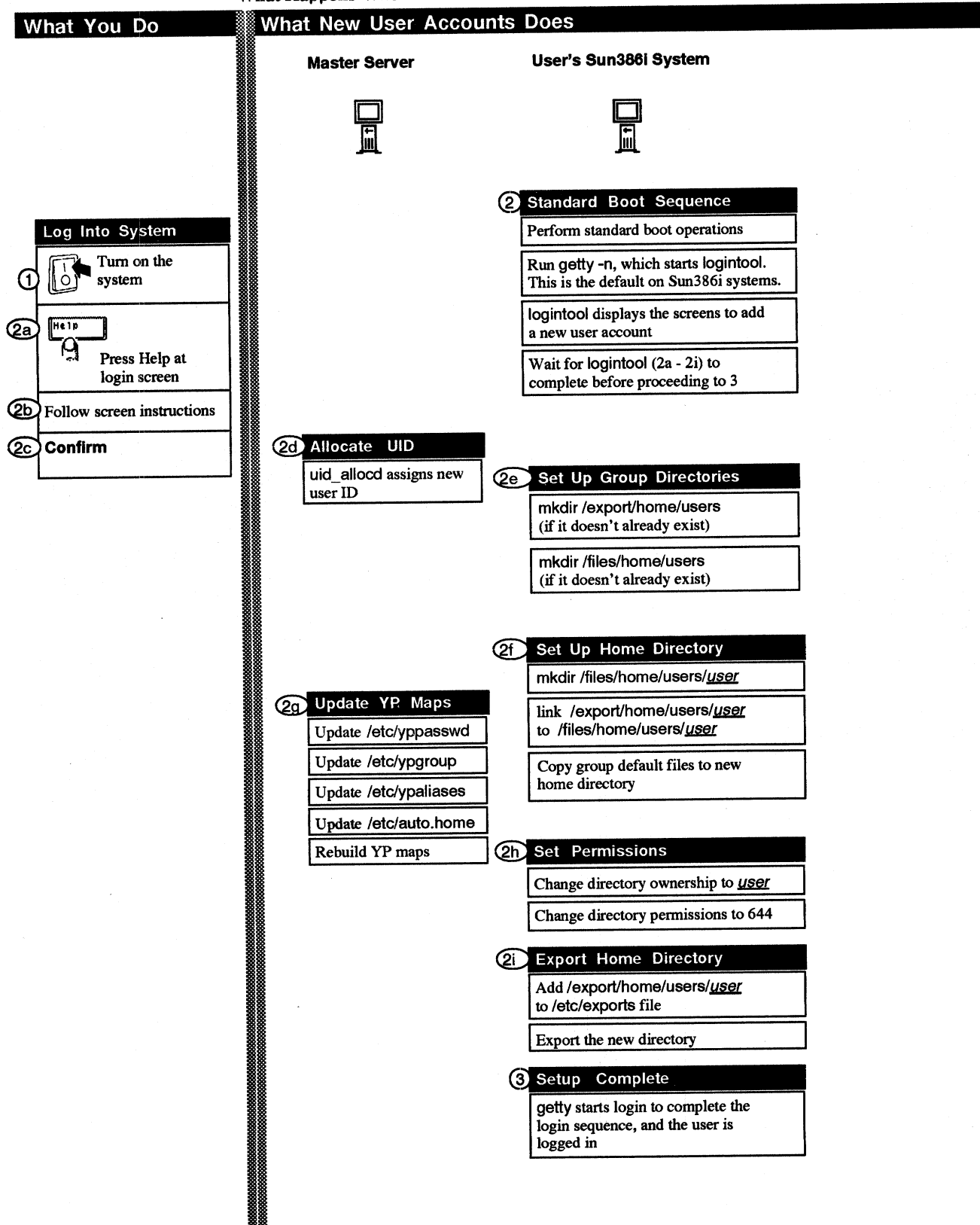
Set Permissions – The user agent daemon sets appropriate directory permissions for the new user.

Export Home Directory – The user agent daemon exports the home directory so that it will be available to all systems in the domain. Once exported, this directory is accessible on the network as `/home/username` via the automounter.

Setup Complete – Control returns to `getty(8)` which runs `login -n` to complete the login process.

- ✓ **man page for `copy_home`** – Sun386i SunOS 4.0 and 4.0.1 do not have a man page for `copy_home`. SunOS 4.0.2 does include a man page on this script.

What Happens When You Add a User via New User Accounts



Adding a Group Through SNAP

The process of adding a group involves two systems:

- ◆ The master server, from which maps are read and updated
- ◆ The group's home directory server, which stores setup files for new users who will be in this group

Groups are treated much as users are: A `yppasswd` entry and home directory are set up for each new group you create. Each group name also serves as a mail alias by which you can address the users who belong to it.

What SNAP Does

SNAP performs the following procedures when you create a new group account. The headings below match those from the diagram.

Read YP Maps – As soon as you select the **Groups** category, SNAP reads the YP maps for the various files listed in the diagram. Files in parentheses () are read but will not be rewritten.

- ✓ Note that SNAP can read files from any YP server (including slave servers). As of 4.0.2, it no longer tries to bind to the YP master.

Allocate UID – Because each group is treated as a user, a new user ID must be assigned. New UIDs are allocated automatically by a daemon (`uid_allocd`) that generates a user ID and then matches it against those listed in the `passwd.byuid` map. The `uid_allocd` daemon also matches this generated UID against its own internal cache of other current UID allocations on the network.

Allocate GID – A unique group ID is assigned by the `gid_allocd` daemon. This procedure is similar to what happens for UID allocation.

Set Up Group Directories – The user agent daemon (`rpc.user_agentd`) creates the appropriate group directory on the group's home directory server.

Set Up Home Directory – Each user group has a home directory containing default files. Although the directory for the default group (`users`) is normally stored on the master server, additional groups can keep their directories anywhere. (The `users` group can also be moved to a different server, if desired, via SNAP.)

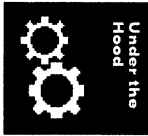
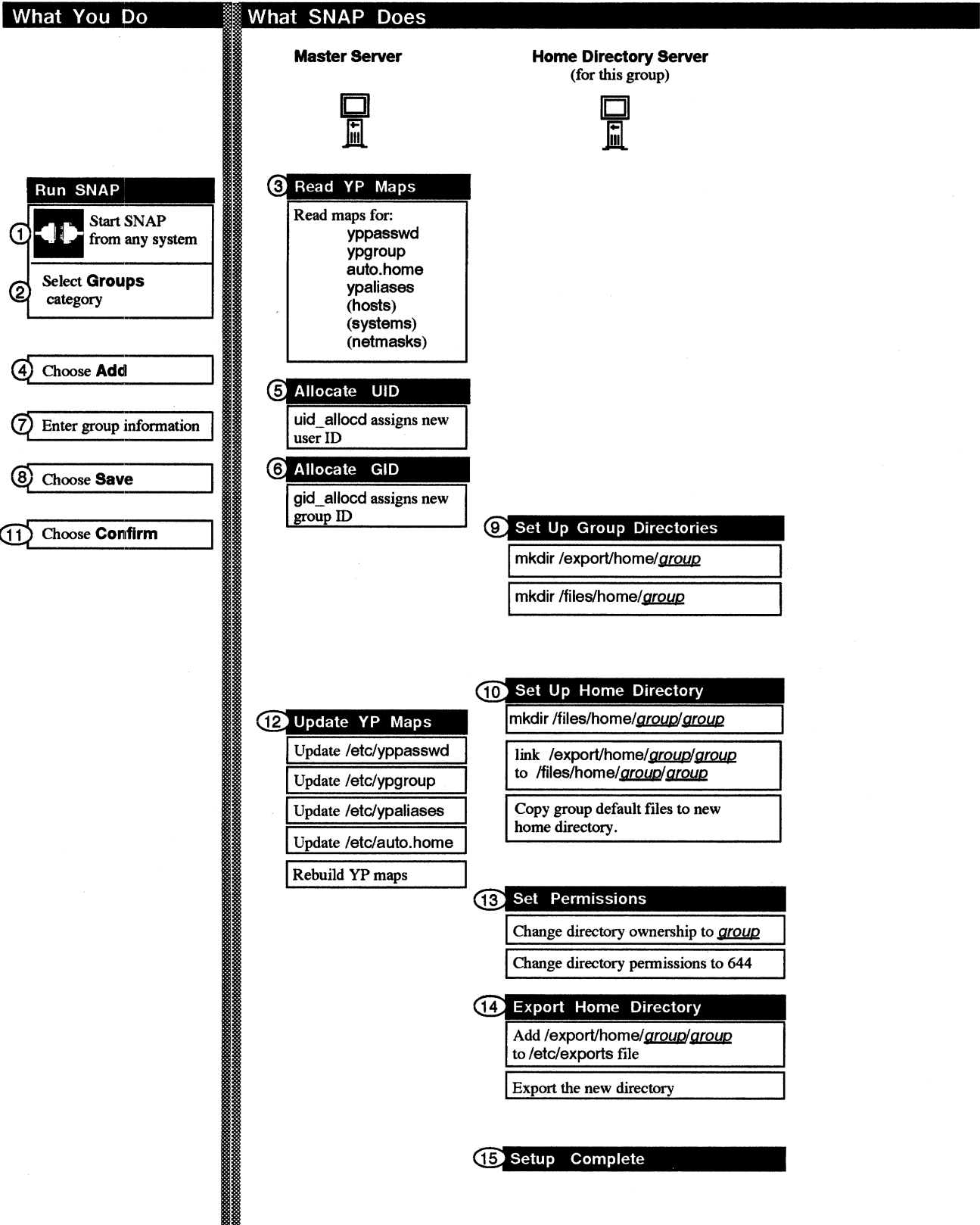
The home directory is set up on the workstation you designate as the "home directory server" for this group. The user agent daemon creates the user's home directory and then sets up a symbolic link so that it can be exported via `/export`. See "Inside the Sun386i File System" at the beginning of this chapter, for more information about exporting conventions.

Also in this step, default files and directories for a new group are copied from the `/home/users/defaults` directory to `/files/home/group/group/defaults`. You can set up custom environments for new users in the group by editing these group default files or adding new ones. If you want the system to perform additional setup tasks during new account creation, add instructions to the `copy_home` script in `/files/home/group/group`.

Update YP Maps – These files are completely rewritten, with additional entries now included for the new group. The same daemon that updates these files (`rpc.ybatchupd`) then re-makes YP, distributing the new maps to all slave servers.

Set Permissions – The user agent daemon sets appropriate directory permissions for the new group.

What Happens When You Add a Group via SNAP



Export Home Directory – The user agent daemon exports the group's home directory so that it will be available to all systems in the domain. Once exported, the automounter makes this accessible on the network as `/home/groupname`.

Setup Complete – SNAP rereads YP maps and is ready for the next operation. As soon as SNAP redisplay its screen, the group account has been set up.

- ✓ **man page for `copy_home`** – Sun386i SunOS 4.0 and 4.0.1 do not have a man page for `copy_home`. SunOS 4.0.2 does include a man page on this script.

Adding a Printer Through SNAP

The process of adding a printer involves two systems:

- ◆ The master server, from which maps are read and updated
- ◆ The print server—the system with the printer

What SNAP Does

SNAP performs the following procedures when you add a printer. The headings below match those from the diagram.

Build Port List – SNAP builds a list of the parallel ports (`/dev/ppn`) and serial ports (`/dev/tty`) that exist on the local system.

Get Valid Printers – SNAP reads a list of existing printers from the `ypprintcap` and `ext_ports` maps, and filters out all entries from `ext_ports` not associated with the local system.

- ✓ **Note** that SNAP can read files from any YP server (including slave servers). As of 4.0.2, it no longer tries to bind to the YP master.

Allocate Printer Name – SNAP allocates a unique printer name by generating `lp` names (`lp`, `lp1`, `lp2`, . . .) until it finds one that does not exist in `ypprintcap`. This name will appear in SNAP as the default name for the new printer.

Make Spool Directory – Creates the `/var/spool/printer` directory to hold waiting print jobs.

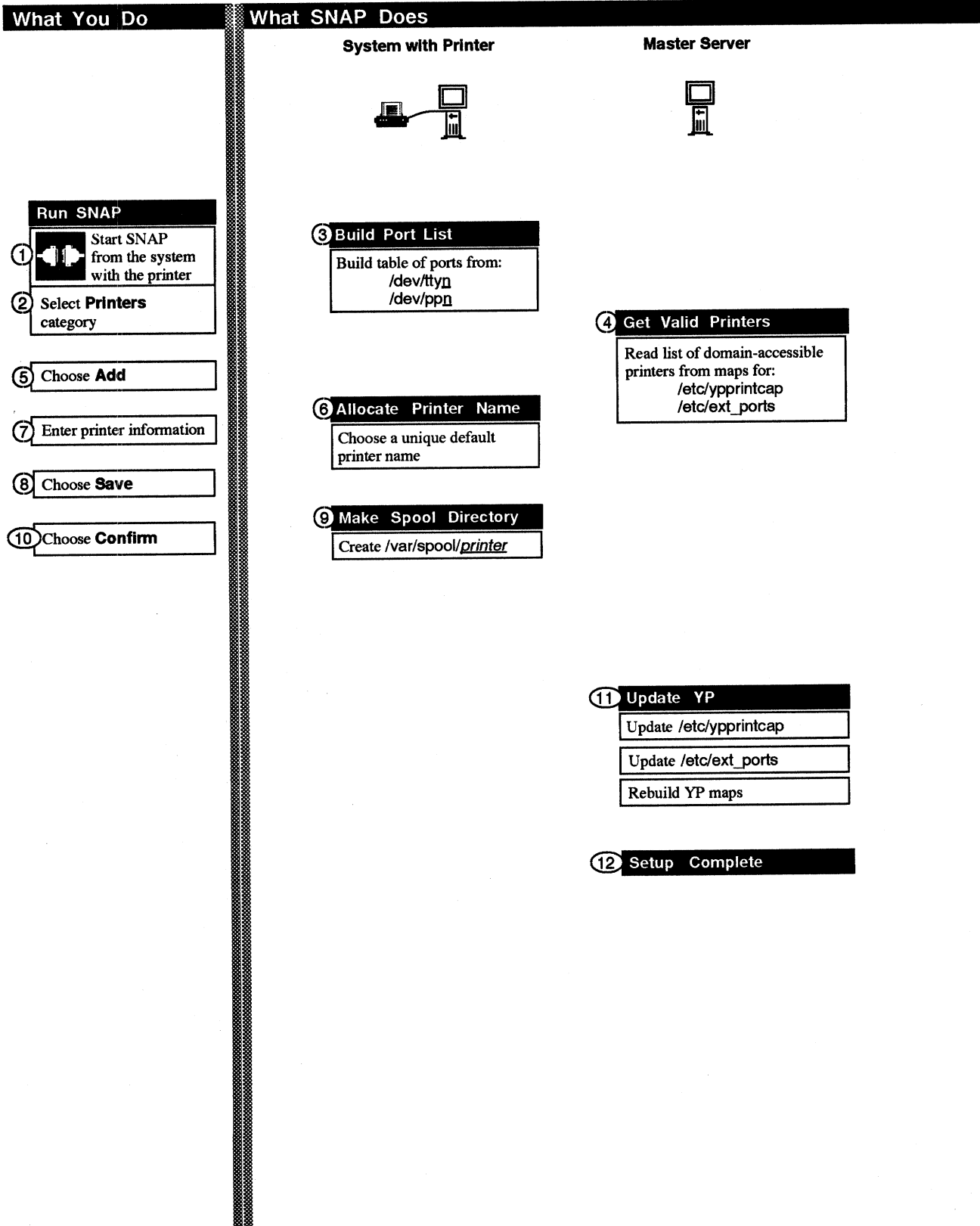
Update YP – The maps are rewritten with updated information. `Printcap` maps now contain the name and location of the new printer, and the `ext_ports` map now knows that this port on the print server is used for printing.

Setup Complete – SNAP rereads YP files and is ready for the next operation. You can begin using the printer from any Sun386i system on the network as soon as SNAP redisplay its screen.

A Note on Local Spool Directories

It is not necessary to create spool directories on other Sun386i machines, because `lpr` and `lpq` create local spool directories as needed. However, if you plan to use this printer from Sun-3, Sun-4, or SPARCstation systems, you'll need to create local spool directories (as `/var/spool/printer`) on those systems.

What Happens When You Add a Printer Using SNAP



Adding a Terminal Through SNAP

The process of adding a terminal involves two systems:

- ◆ The master server, from which the `ext_ports` map is read and updated
- ◆ The system with the terminal

What SNAP Does

SNAP performs the following procedures when you add a terminal. The headings below match those from the diagram.

Build Port Table – SNAP builds a list of the serial ports (`/dev/tty*`) that exist on the local system.

Get Valid Port List – SNAP reads a list from the `ext_ports` map of existing ports and their current assigned uses. It also filters out all entries from `ext_ports` that are not associated with the local system.

Set Port Permissions – When you enable the terminal, SNAP sets the port permissions to 622 so that only the `getty` daemon can open it. This prevents other applications (such as DOS) from accessing the serial port until it is disabled. When you disable or remove the port, SNAP sets the permissions back to 666 so that anyone can read or write to the port.

- ✓ In Sun386i 4.0.2, permissions are reset to 666 when you disable or remove the terminal in SNAP (a bug in Sun386i 4.0.1 prevented this).

Modify ttytab – Add the `getty(8)` and terminal type information to the appropriate line in `/etc/ttytab(5)`.

Modify ext_ports – Note the current use of this port in `/etc/ext_ports`, and rebuild YP.

Enable Soft Carrier – Instruct the kernel to ignore the carrier detect line.

Background: For a modem, SunOS systems use a mode called “hard carrier detect,” which basically waits for the serial port’s carrier detect line to be asserted before opening the port for use. Because terminals do not need to wait for a carrier, they can ignore this serial line. On Sun386i systems, “soft carrier” mode is set for terminals so that the port is opened regardless of the carrier line’s current state.

Setting this soft carrier mode has the same effect as the traditional procedure of disabling carrier detect in the kernel and then rebuilding it. SNAP sets soft carrier mode through an I/O control. Users can do the same thing manually by issuing the following command:

```
system: SUPERUSER:1} /usr/etc/ttysoftcar -y devicename
```

Start getty Daemon – SNAP sends a signal to `init` so that it rereads `/etc/ttytab` and starts the `getty` daemon running on the specified port.

Setup Complete – SNAP rereads YP files and is ready for the next operation. You can begin using the terminal as soon as SNAP redisplay its screen.

What Happens When You Add a Terminal Using SNAP

What You Do

What SNAP Does

System with Terminal

Master Server



Run SNAP

- 1 Start SNAP from the system with the terminal
- 2 Select **Terminals** category
- 5 Choose **Add** & enter information
- 6 Choose **Save**
- 8 Choose **Confirm**

3 Build Port Table

Build table of ports from /dev/tty

4 Get Valid Port List

Read ext_ports map and extract list of available ports for the system in question

7 Set Port Permissions

chmod *mode* /dev/tty

9 Modify ttytab

Update /etc/ttytab with appropriate terminal settings

10 Modify ext_ports

Update /etc/ext_ports so the use of this port is recorded

Rebuild ext_ports map

11 Enable Soft Carrier

Issue IOCTL to set soft carrier mode in kernel

12 Start getty Daemon

Restart init process (kill -1 1) so that it reads ttytab and starts the getty daemon

13 Setup Complete



Adding a Modem via SNAP

The process of adding a modem involves two systems:

- ◆ The master server, from which the `ext_ports` map is read and updated
- ◆ The system with the modem

What SNAP Does

SNAP performs the following procedures when you add a modem. The headings below match those from the diagram.

Build Port Table – SNAP builds a list of the serial ports (`/dev/tty*`) that exist on the local system.

Get Valid Port List – SNAP reads a list from the `ext_ports` map of existing ports and their current assigned uses. It also filters out all entries from `ext_ports` that are not associated with the local system.

Set Port Permissions – When you enable the modem for either dial-in or dial-out, SNAP sets the port permissions to 622 so that only the `getty` daemon can open the port. This prevents other applications (such as DOS) from accessing the serial port until it is disabled. When you disable the modem for both dial-in and dial-out, SNAP sets the permissions back to 666 so that anyone can read or write to the port.

- ✓ In Sun386i 4.0.2, permissions are reset to 666 when you disable or remove the modem in SNAP (a bug in Sun386i 4.0.1 prevented this).

Set Up Dialer – SNAP creates an alternate device for the modem using `mknod`. The device name begins with `cu` and corresponds to the port ID (letter/number) of the serial device to which the modem is attached. Examples:

```
/dev/cua for /dev/ttya
```

```
/dev/cum0 for /dev/ttym0
```

```
/dev/cum1 for /dev/ttym1
```

Permissions on the `/dev/cuportID` device are set at 600 for dial-out or dial-in/dial-out and 000 for dial-in-only (no dial-out) modems.

Modify Local Files – Update `/etc/ttytab` and `/etc/remote` with the appropriate modem entries.

Modify `ext_ports` – Record the current use of this port in `/etc/ext_ports`, and rebuild YP.

Disable Soft Carrier – Instruct the kernel to acknowledge the carrier detect line.

Background: For a modem, SunOS systems use a mode called “hard carrier detect,” which basically waits for a serial port’s carrier detect line to be asserted before opening the port for use. The “soft carrier” mode, normally set for terminals, must therefore be disabled when you’re using a modem on Sun386i systems.

Disabling this soft carrier mode has the same effect as the traditional procedure of enabling carrier detect in the kernel and then rebuilding it. SNAP controls soft carrier mode through an I/O control. Users can do the same thing manually by issuing the command:


```
/usr/etc/ttysoftcar -n devicename. See Chapter 1 for further information.
```

Start `getty` Daemon – SNAP sends a signal to `init` so that it rereads `/etc/ttytab` and starts the `getty` daemon running on the specified port.

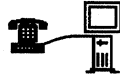
Setup Complete – SNAP rereads YP files and is ready for the next operation. You can begin using the modem as soon as SNAP redisplay its screen.

What Happens When You Add a Modem Using SNAP

What You Do | What SNAP Does

- Run SNAP**
- 1  Start SNAP from the system with the modem
 - 2 Select **Modems** category
 - 5 Choose **Add** & enter information
 - 6 Choose **Save**
 - 9 Choose **Confirm**

System with Modem



- 3 **Build Port Table**
Build table of ports from `/dev/tty \underline{a}`
- 7 **Set Port Permissions**
`chmod mode /dev/tty \underline{a}`
- 8 **Set Up Dialer**
Create `/dev/cu \underline{portID}`
Set `/dev/cu \underline{portID}` permissions
- 10 **Modify Local Files**
Update `/etc/ttytab`
Update `/etc/remote`
- 12 **Disable Soft Carrier**
Issue `IOCTL` to disable soft carrier mode in kernel
- 13 **Start getty Daemon**
Restart `init` process (`kill -1 1`) so that it reads `ttytab` and starts the `getty` daemon
- 14 **Setup Complete**

Master Server



- 4 **Get Valid Port List**
Read `ext_ports` map and extract list of available ports for the system in question
- 11 **Modify ext_ports**
Update `/etc/ext_ports`
Rebuild YP map



Backing Up Data Through SNAP

SNAP backups can involve several systems on the network:

- ◆ The local system that will perform the backup
- ◆ Your home directory server, where a catalog of backup information is stored
- ◆ The file servers containing the files you plan to back up

Backing up files is actually a two part process. In Part A, you schedule the backup via SNAP. In Part B, shown on the pages following, `cron` starts a job at the appointed time to perform the backup.

Part A – What SNAP Does

Build Table of Backups – All backups on a machine—scheduled or unscheduled—are recorded in `root`'s `crontab(5)` file. Each backup has its own ID which is used to determine file permissions and to access backup catalog files and maps.

SNAP examines the unique backup IDs, backup names, and other flags for each backup line in `root`'s `crontab` file. SNAP translates this information into the backup list you see in the bottom SNAP panel.

Start Organizer – If you ask to back up selected files, SNAP starts a special version of Organizer. You use this Organizer window to choose the files to back up.

Write Backup List – When you choose the **Backup** button in Organizer, it writes a list of selected files to `/var/spool/backup/BACKUP_unique_id`. A typical name for the file list might be:

```
BACKUP_425776879_822089534
```

The program `/usr/etc/backup` will pass this file list to `bar` in a later step.

Update `root` `crontab` – As soon as you choose **Sched** to schedule the backup, SNAP sets up this backup in `root`'s `crontab`. If you've asked to be reminded before the backup, SNAP will also create a separate `crontab` line to remind you of the backup via the console window, mail, or both.

Finally, SNAP writes to the named pipe `/var/spool/cron/FIFO` to alert `cron(8)` to the new jobs pending.

Setup Complete – SNAP rereads the `root` `crontab` file and is ready for the next operation.

What Happens When You Use SNAP Backup

What You Do Part A – What SNAP Does

Your System



Your Home Directory Server



Run SNAP

1 Start SNAP from your system

2 Select **Backup** category

4 Choose **Setup**

5 Enter backup profile information

6 Choose **Select Files** (optional)

8 Select files in Organizer

9 Choose **Backup** in Organizer window

10 Choose **Done** in Organizer window

12 Move back to SNAP window

13 Choose **Sched** to schedule the backup

14 Choose **Confirm**

3 Build Table of Backups

Read root crontab file and extract all backup lines

Examine flags to determine ownership and whether backup is scheduled

7 Start Organizer

Start a special version of Organizer

11 Write Backup List

Write the list of selected files to `/var/spool/backup/BACKUP_unique_id`

15 Update Root crontab

Write backup information to `/var/spool/cron/crontab/root`

Schedule reminder in this same crontab file

Write to named pipe in `/var/spool/cron` to schedule new cron jobs

16 Setup Complete



Part B – What cron Does

The program `/usr/etc/backup` is responsible for both initiating backups and for reminding you about backups that are scheduled to take place.

Send Reminder – If you have asked to be reminded about the backup, `cron` executes a line such as the following, and you will receive notice of the backup via the console window, mail, or both:

```
00 16 * * * /usr/etc/backup T BACKUP_425776879_822089534 "My Backup" Full /dev/rfd0a Full Both Once mtravis 5075 0
```

Start Backup – A separate `crontab` line initiates the backup at the appointed time:

```
00 23 * * * /usr/etc/backup B BACKUP_425776879_822089534 "My Backup" Full /dev/rfd0a Full Both Once mtravis 5075 0
```

The `/usr/etc/backup` program reads the appropriate files in `/var/spool/backup` (the file is named `BACKUP_425776879_822089534` in this example) and builds a list of files and directories to be backed up. It then passes this list to `bar`, which performs the actual backups.

Done – When the backup is done, remove the tape and label it.

What Happens When You Use SNAP Backup (cont'd)

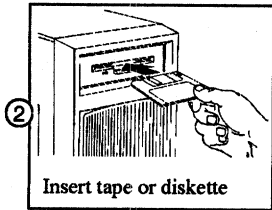
What You Do | **Part B - What cron Does**

Your System

Your Home Directory Server



1 Send Reminder
Mail or display backup reminder



2

Insert tape or diskette

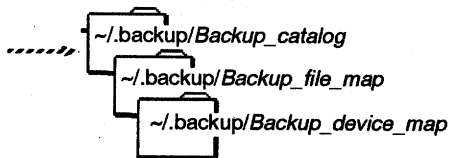
3

Wait for backup

4 Start Backup
Start /usr/etc/backup at the appointed time
Write to backup catalog
Back up files via bar
Mail or display backup completion message

4a

Insert media as needed



6

Remove tape or diskette and label it

5 Done



What's in the crontab Entry

Here is a typical crontab line as written by SNAP Backup:

```
00 16 *** /usr/etc/backup T BACKUP_425776879_822089534 "My Backup" Full /dev/rfd0a Full Screen Once mtravis 5075 0
```

Diagram labels for the crontab line above:

- 00 16 ***: cron time entries
- /usr/etc/backup: backup command
- T: Backup flag
- BACKUP_425776879_822089534: File list
- "My Backup": Backup description
- Full: Type
- /dev/rfd0a: Device
- Full Screen: (not used) / Notification method
- Once: How often?
- mtravis 5075 0: User name and ID

cron time entries – Standard cron time settings. See `crontab(1)` or Sun386i Advanced Skills for details.

backup command – This command starts backups, notifies you of pending backups or errors, and keeps logs of backups performed.

Backup flag – Valid entries are: B for a scheduled backup, xB for an unscheduled backup, R for a running backup, T for a backup reminder, and xT for an unscheduled backup reminder.

File list – The name of the file in `/var/spool/backup` that contains a list of the files to back up.

Notification method – You have the choice in SNAP of Screen, Mail, or Both.

Files flag – Indicates what files are to be backed up: 0 is for all home directory files, 1 is for specific home directory files, 2 is for all files on the local system, 3 is for specific files on the local system, 4 is for all files on the network, and 5 is for specific network files.

Backup description, Type, Device, How often?, User name and ID – Additional information used when backing up files or notifying you about backups.

Options used by `/usr/etc/backup`

The `bar` command options used by `/usr/etc/backup` when backing up files are as follows:

f – followed by device name, `/dev/rst8` or `/dev/rfd0a`

v – for verbose output, used to create the backup catalog

K – to follow directory symbolic links specified on the command line

H – followed by the backup description; this is used in the `bar` volume header

I – followed by the name of the file in `/var/spool/backup` that contains the list of files to be backed up

Z – to have `bar` compress files when archiving

U – followed by the user ID of the user that specified the backup; this is used in the `bar` volume header

M – if the backup is to be incremental, followed by the cut-off date for the backup

D – followed by the date and time of the backup; this is used in the `bar` header

N – to force `bar` to check the ownership of the media before writing

Continuing a Backup

If `bar` requires an additional tape or diskette, a child process of `/usr/etc/backup` sends a message to you via mail or the console window. When you insert a new tape or diskette and then choose **Continue** in SNAP, SNAP sends a signal to this child process, and the child instructs `bar` to continue backing up onto the next volume.

Backup and the operator Group

A member of the `operator` group is given the following privileges:

- ◆ root file access in Organizer when selecting files for backup or restore
- ◆ Option of backing up all system or all network files
- ◆ Ability to unschedule or remove another user's backup profiles
- ◆ Ability to restore files from another user's backup



Restoring Data through SNAP

SNAP Restore can involve two or more systems on the network:

- ◆ Your home directory server, where a catalog of backup information is stored
- ◆ The file servers on which you will restore your data

What SNAP Does

SNAP performs the following procedures when restoring files. The headings below match those from the diagram.

Start Organizer – SNAP starts a special version of Organizer that you use to choose the files to be restored. To build the file list, Organizer reads the backup catalog in your home directory. If this catalog does not exist, you can recreate it from the backup tape or diskette using the **Add Entry** button in SNAP. (Note that if you choose **Restore All**, SNAP Restore does not read the backup catalog files.)

If you have previously selected files to restore from this backup, Organizer reads this list of selected files from `/var/spool/backup/RESTORE_uniqueID` and highlights those files.

Build Restore List – SNAP builds a temporary list of files to be restored in `/var/spool/backup`.

Restore Files – SNAP starts `bar`, which uses the following options to restore from the backup:

`f` – followed by device name, `/dev/rst8` or `/dev/rfd0a`

`v` – for verbose output

`I` – followed by the name of the file in `/var/spool/backup` that contains the list of files to be restored

`Z` – to have `bar` uncompress files when extracting

`T` – to stop when the first occurrence of every file is found

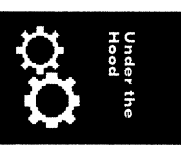
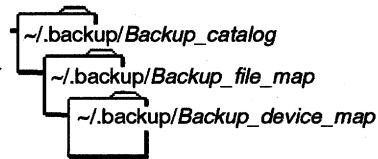
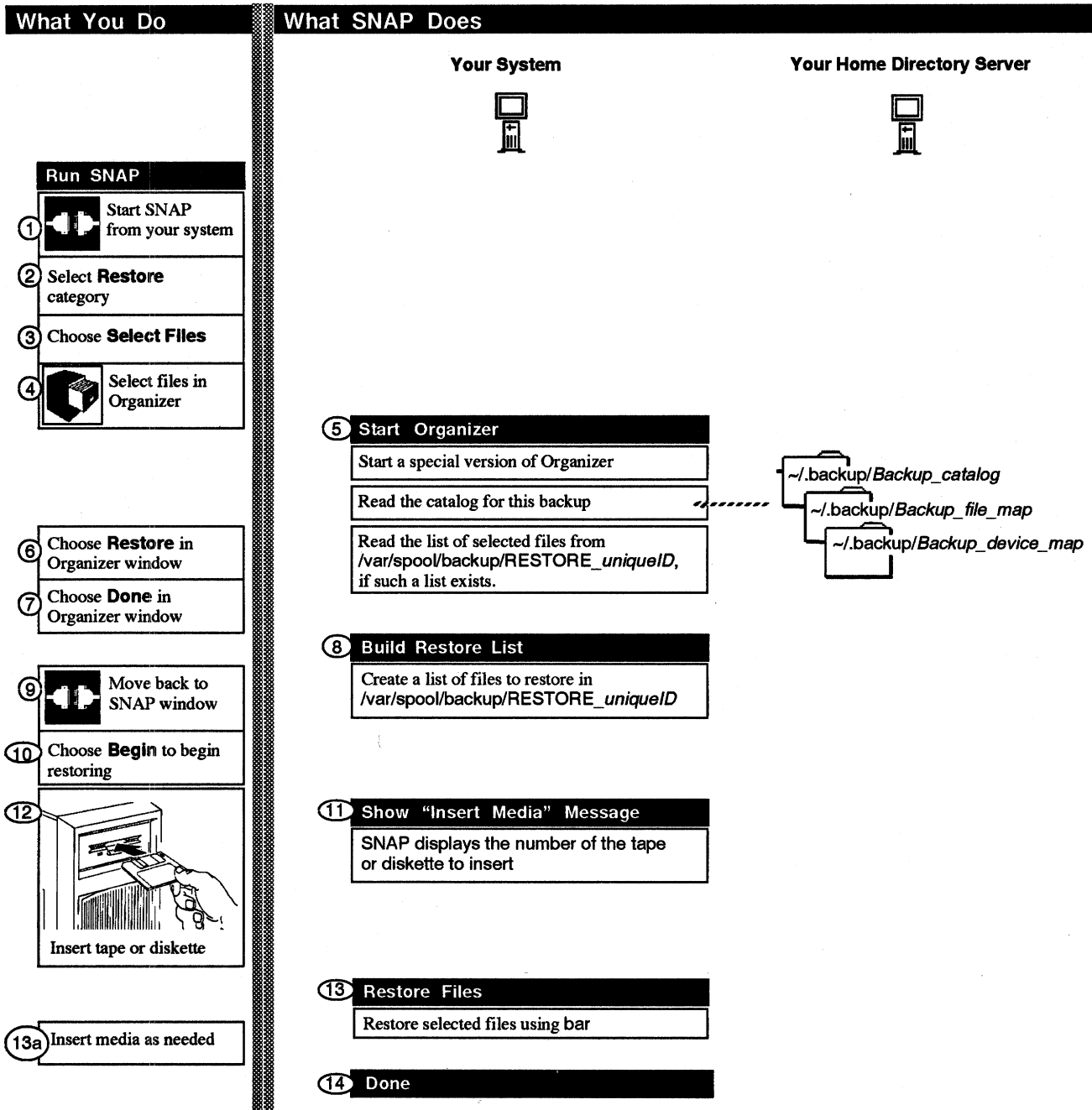
`S` – to restore files to a new directory, if specified

`s, U, G` – to restore files to a new user, if so specified

`O` – to check the media for ownership, if not in the `operator` group

Done – Restoration of the backup is complete.

What Happens When You Use SNAP Restore



Booting a Network Client

The boot procedure on a Sun386i is different than on other Sun workstations in the following ways:

- ◆ Sun386i workstations normally display a minimum amount of information when they boot. Other Sun workstations display more information that is meaningful to SunOS experts but not to most users. However, you can have the additional information displayed on a Sun386i system by changing an NVRAM value. (See Chapter 4 for details.)
- ◆ Sun386i network clients, diskless clients, or diskful clients automatically get the information necessary for installation and booting from the network. For other Sun workstations, the information must be resident on the system and must be manually configured at installation.
- ◆ Sun386i workstations store network configuration information in a file (`/etc/net.conf`) that is unique to Sun386i systems. Other Sun workstations store the information in several files.
- ◆ A Sun386i workstation has identical run command files (`/etc/rc`, `/etc/rc.boot`, and `/etc/rc.local`) regardless of its role on the network. For other Sun workstations, the contents of these files must be modified based on the system's network role.

The headings below match those from the diagram.

Start standard boot sequence – The Sun386i network client begins by following the standard boot sequence used on all Sun systems: It starts with `boot(8S)`, which starts `init(8S)`, which starts a shell to execute `/etc/rc.boot`. The `rc.boot(5)` script begins by performing a file system check, just as on other Sun systems. Once the file systems have been checked, `rc.boot` proceeds to examine the system's `/etc/net.conf` file (next step).

Execute `/etc/net.conf` – The `net.conf` file contains some basic information about the system. Here is an example of a `net.conf` file:

```
HOSTNAME=hostname
DOMAINNAME=YP.noname.com
NETWORKED=yes
PNP=yes
VERBOSE=no
```

When this file is executed, each variable and its setting is read into memory. Note that some of these variables may be reset in later steps.

Configuration probing on? – The setting for `PNP` determines whether the system will do configuration probing (next step). If configuration probing is enabled (`PNP=yes`), the system will probe the network for configuration information in later steps, and may reset some of the variables in `/etc/net.conf`. If configuration probing is disabled (`PNP=no`) then the system will use the `/etc/net.conf` settings as is.

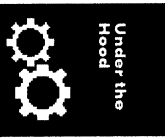
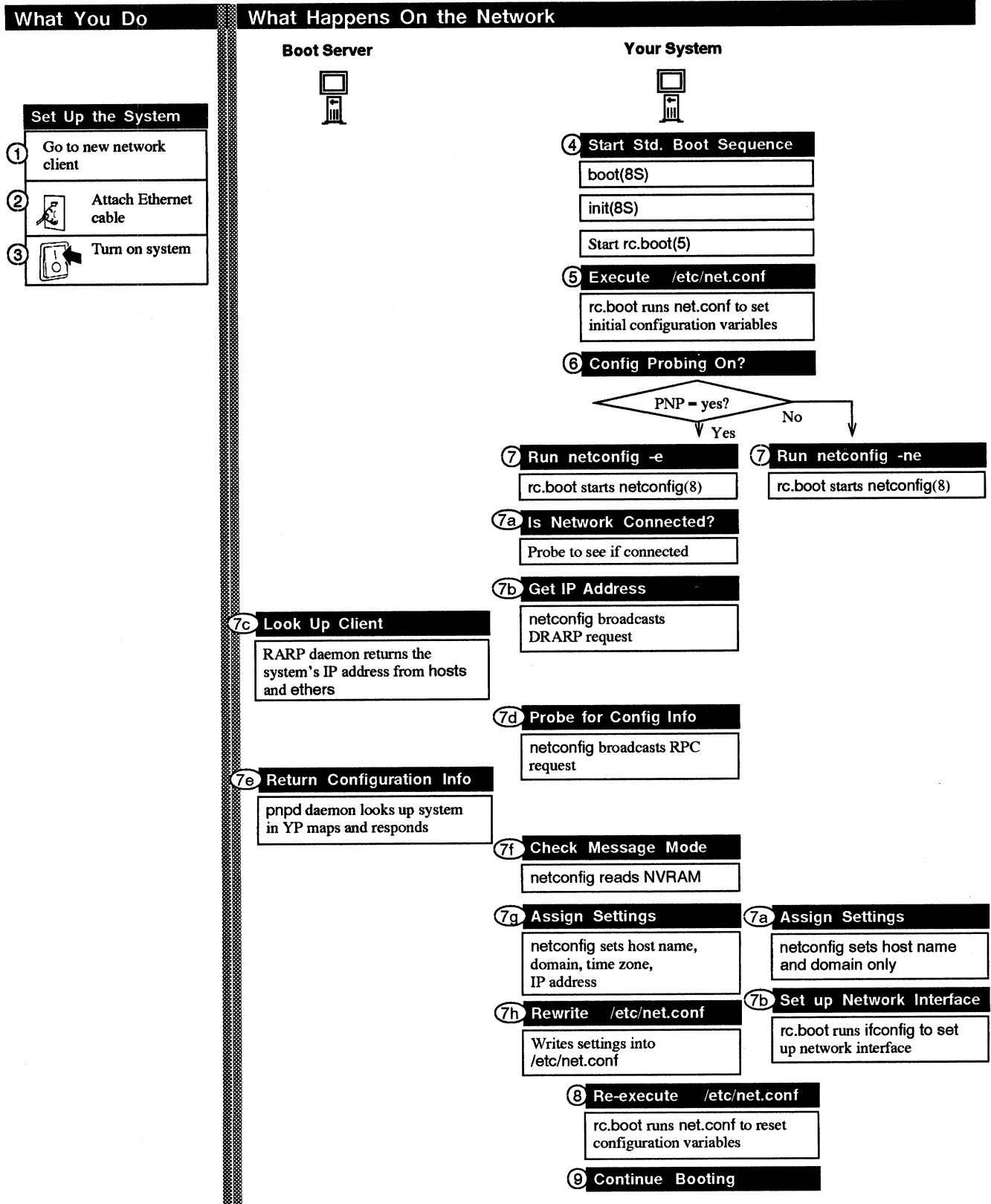
Run `netconfig` – The program `/sbin/netconfig` is started from `rc.boot`.

If `PNP=yes`, then `rc.boot` runs `netconfig -e`, which checks the `HOSTNAME` and `DOMAINNAME` variables against the `ypservers` map to see if this is a master or slave server. If the system is not a server, `netconfig` commences with its IP address and configuration requests.

If `PNP=no`, the system runs `netconfig -ne`. The `-n` option skips configuration probing.

Is Network Connected? – `netconfig` probes to see if the system is connected to the network.

What Happens When You Boot a Network Client



Get IP address (Done only if configuration probing is enabled) – The program `/sbin/netconfig`, started from `/etc/rc.boot`, broadcasts a Dynamic RARP (DRARP) request for the IP address of this system.

Look up client (Done only if configuration probing is enabled) – The first server (master or slave) to answer the RARP request becomes the temporary boot server that will provide the IP address to the new client.

The `rarpd` daemon running on the boot server uses the client's Ethernet address as a key to look up the system's assigned IP address. The daemon first matches the Ethernet address with an entry in the YP `ethers` map to determine the client's host name. The daemon then uses this host name to look up the client's IP address in the YP `hosts` map.

The IP address is then returned to the client.

Probe for configuration info (Done only if configuration probing is enabled) – Once the `netconfig` program knows the IP address, it sends out an RPC request for additional configuration information, such as the name of the domain and the host name. Again, the first server to answer becomes the temporary boot server that manages the configuration information.

Return configuration info (Done only if configuration probing is enabled) – The `pnpd` daemon on the boot server looks up configuration information in the YP maps, and passes it back to `netconfig` on the client system.

Check message mode (Done only if configuration probing is enabled) – `netconfig` checks the value at NVRAM location 494. If the value at this location is 0, `netconfig` disables explicit boot messages by setting `VERBOSE=no`. If the value at location 494 is 1 or 2, `netconfig` sets `VERBOSE=yes`.

Assign settings – If configuration probing is enabled, then `netconfig` sets the host name, domain name, time zone, and IP address.

If configuration probing is disabled, `netconfig` sets the host name and domain name (based on the original values set from `/etc/net.conf`), but it does not set the time zone or IP address. The system's IP address is instead determined from its local `/etc/hosts` file.

Set up Network Interface (Done only if configuration probing is disabled) – `rc.boot` runs `ifconfig(8c)` to configure network interface parameters.

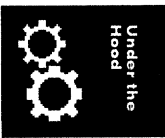
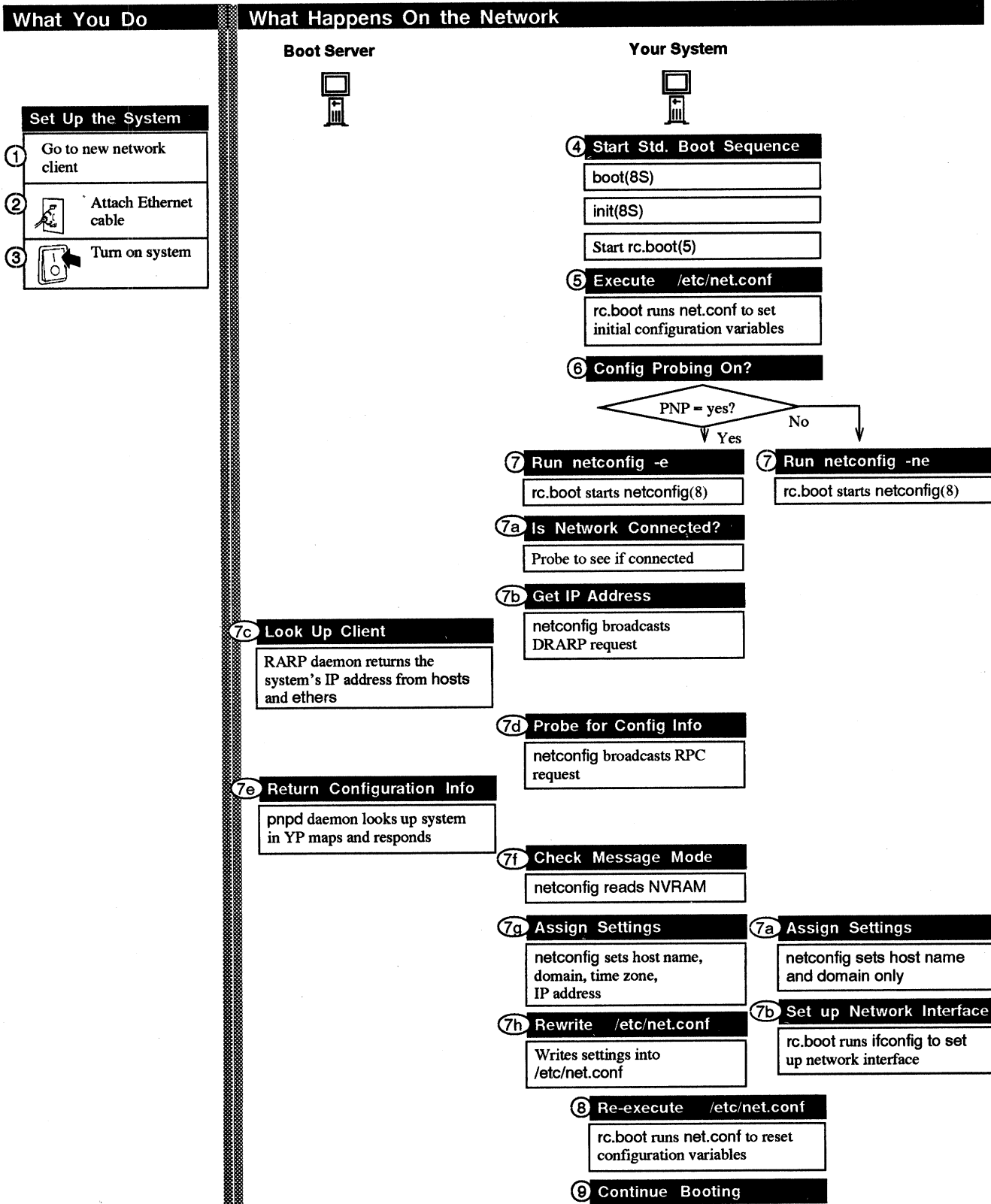
Rewrite `/etc/net.conf` (Done only if configuration probing is enabled) – All variables are rewritten to the `/etc/net.conf` file. Some of these values may have been changed during configuration probing.

Re-execute `/etc/net.conf` – `rc.boot` executes this file again so that any new variable settings take effect. Although this step is not strictly necessary in cases where `PNP=no`, re-executing the file simplifies the `rc.boot` script.

Continue booting – The system finishes running `rc.boot`, and then runs `rc` and `rc.local`.

Reference: Chapter 5 of this manual (“Sun386i Probing” section)
Chapter 10 of this manual (`net.conf` description)
Sun386i Advanced Administration, (February 1989 edition, Chapter 1)
On-line Sun386i SunOS 4.0.2 man pages (`man_pages` optional cluster must be loaded) — `netconfig` (8S), `rc.boot` (8S)

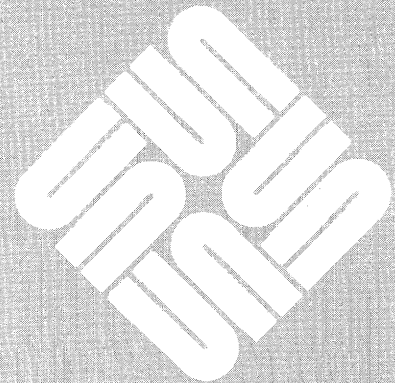
What Happens When You Boot a Network Client
 (This diagram is identical to the one presented on page 171.)



Notes

HINTS AND TIPS

HINTS AND TIPS	1427
Sun386i Backups	1427
DOS Memory Tips	1429



HINTS AND TIPS

Sun386i Backups

Sun386i Backup Hints

Customers may find that regularly scheduled backups which used to take from 15 minutes to two hours when running Sun386i SunOS 4.0.1 may now take increasing amounts of time after upgrading to Sun386i SunOS 4.0.2. This article contains the problem definition and two hints to reduce time needed for backups.

The Problem Defined

SNAP backup writes the list of files backed up to the backup catalog, which resides in the `.backup` subdirectory of the user's home directory. The backup catalog is composed of the three files listed below.

`Backup_device_map`
maps device names to unique IDs

`Backup_file_map`
maps filenames to unique IDs

`Backup_catalog`
contains the backup info (fileid, date, time, description, deviceid, and volume number)

The backup catalog is used when you press the **SELECT FILES** button in SNAP's restore category. A mini-organizer displays the backup catalog and allows you to selectively chose files to restore.

Because the backup catalog is composed of simple ASCII files, searches and sorts of those files will take longer as the files grow.

Hint 1: Check File Size

You should check the size of the above listed files, and purge entries by date using the **REMOVE ENTRY** button in SNAP's restore category. Note that **REMOVE ENTRY** changes the ownership of the backup catalog files to root, so you will have to use `chown(1)` after doing the remove entry.

Hint 2: Remove Entries
Manually

Alternately, you can remove every entry in the above listed files manually. After doing this you will no longer be able to selectively chose files to restore through **SELECT FILES** in SNAP's restore category.

Individual files can be restored using `bar` as shown below.

```
% bar xvfzp /dev/rfd0a file1 file2 ... /* for floppy */  
% bar xvfzp /dev/rst8 file1 file2 ... /* for tape */
```

RESTORE ALL in SNAP's restore category does not use the backup catalog.

Fixing the bug in Sun386i SunOS 4.0.1 that caused the backup catalog to get corrupted now causes SNAP backups to take longer in Sun386i SunOS 4.0.2 because the backups write to the backup catalog more often during the backup.

Keeping the size of the catalog small should help with the backup performance.

DOS Memory Tips

DOS Memory Usage Shooting Tips

This article contains DOS memory usage tips for those running Sun386i SunOS 4.0.1 or 4.0.2 on Sun386i workstations.⁴

Tip 1: Determining Available Memory

Not all applications know how to use the expanded memory driver. Some applications can use the expanded memory driver but do not auto-detect its presence. These applications may require a different configuration or switch settings or both when started.

You can determine how much memory is available for a application in the DOS window by using the `chkdsk c:` command. This will report the amount of free memory as shown in the below example. The example shows there are '546144 bytes free'.

```
D:\HOME\EDEAN\DOSAPPS>chkdsk c:

21309440 bytes total disk space
  55296 bytes in 2 hidden files
  12288 bytes in 6 directories
 2580480 bytes in 166 user files
18661376 bytes available on disk

655360 bytes total memory
546144 bytes free
```

```
D:\HOME\EDEAN\DOSAPPS>
```

The Sun386i DOS window uses more memory for resident drivers than most PCs due to the amount of emulation that we need to load. A big difference can usually be found by removing the redirector which is uses about 50k of memory for Sun386i SunOS 4.0.1, or about 25.9k for those running Sun386i SunOS 4.0.2.

Of course by removing this option you would loose all networked drives. You can do a quick test by turning off the redirector by editing your `C:autoexec.bat` file by commenting-out the redirector entry, and then rebooting the DOS window.

In order to address the problem correctly, you need to know just how much memory is available on the XT, AT, and on the Sun386i using the `chkdsk` command. Very often a DOS application can be configured to use less memory.

⁴ The tips contained in this article are submitted by Ed Dean, Sun BOS Software, Boston Development Center.

Note that if you are comparing an application's performance on an XT or AT to a Sun386i, remember that XTs and ATs do not have configurations similar to the default Sun386i configuration. For example, most XT and AT machines do not have redirected drives installed.

Tip 2: DOS Extenders

If you find that an application is too large, and there is *not* a memory emulation problem, consider using a DOS extender, available from third-party vendors. One example is the QEMM386, runs in v86mode, and is available from QuarterDeck.⁵

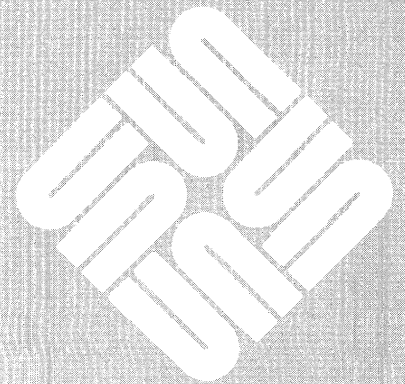
The extender provides the user with a transient program area close to the full 640k by moving drivers, redirectors, and the like to expanded memory. For example, PC LAN software typically uses 50-70k, and you can load this into 'high memory'.

⁵ Please note that this reference is not an endorsement of the example product named. Customers may wish to explore other DOS extenders as well.

THE HACKERS' CORNER

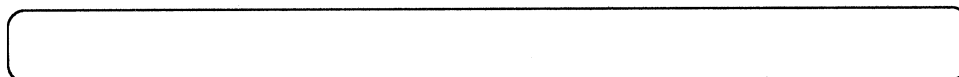
THE HACKERS' CORNER 1433

FreeSpace 1433



THE HACKERS' CORNER

FreeSpace



FreeSpace, the Final Frontier

This month's **Hackers' Corner** contains **FreeSpace**, a tool to help you in your disk space, cost-saving voyages. It is an ongoing mission, to explore efficient new ways of cleaning up old files, to seek out ways of deleting them, to boldly delete where no one has deleted before....⁶

The below tool was written to help find large, old files that are no longer needed, so you can delete them and free new frontiers of disk space. If each user deleted just one Mbyte of unneeded files, it would save your company unnecessary costs by postponing the need to add more disks.

Please consult your local shell script or programming expert regarding any script or code problems. The example programs are not offered as a supported Sun product, but as items of interest to enthusiasts wanting to try out something for themselves. Note that **Hackers' Corner** code may not work in all cases, and may not be compatible with future SunOS releases.

Using the FreeSpace Tool

Follow the steps listed below to use **FreeSpace**.

1. Save the below code to a file named `cldir`.
2. Enter the command `chmod +x cldir`.
3. Enter the command `cldir`. The sort selection menu will appear as shown below.
4. Type `h` if you need help. The action options menu will appear as shown below.

⁶ This month's **Hackers' Corner** is submitted by Bill Petro, GSG Marketing, Mountain View, California, USA.

An Example Usage

An example usage is shown next which specifies sorting files by size, followed by skipping over the largest file, selecting the help screen, and then quitting FreeSpace.

```
Be Happy :- )cldir
Sort Selection:
```

```

a - Alpha, alphabetical
A - All Alphabetical, including dot files
p - Pattern, by pattern with wildcards
r - Reverse date
s - Size, largest first
L - Larger than, over xx bytes
t - Time, oldest first
T - Age, xx days ago
q - Quit
```

```
Enter letter of choice: s
```

```
Sorting by size ...
```

```
Use (n)ext, (l)ist, (p)rint, (d)elete, (q)uit, e(x)it, or (h)elp
```

```
386i.index.awk: 42272 Aug 28 15:11 = ascii text : [nlpdqx]:n
```

```
windows.detail: 39475 Sep 13 12:49 = [nt]roff, tbl, or eqn input te: [nlpdqx]:h
```

```
Action options:
```

```

n          next
l or .    more
p          print
d or r    deleting
f          Files in sort selection
M          mail
:          single csh command
!          new csh shell
q          quit this level
x          exit
```

```
windows.detail: 39475 Sep 13 12:49 = [nt]roff, tbl, or eqn input te: [nlpdqx]:q
```

```
Exiting ...
```

```
Be Happy :- )
```

SunOS Revision Levels

The below tool works for those running SunOS 4.x, since the `/bin/find` is used. For those running SunOS 3.x, you will need to change this command pathname to `/usr/bin/find`.

The FreeSpace Tool

The code for the FreeSpace tool appears on the following pages.

For an online copy of the FreeSpace code, simply send email to *sun!stb-editor*. Please specify that you want the November 1989 Hackers' Corner code.

```

#!/bin/sh -
# cldir - present a list of sorted files for display, printing or deleting
#
# This tool is intended to be used to clean up a directory, including
# it's subdirectories. It presents a list of files that the user
# may then display, print, delete, examine with a mail program, etc.
# The script can be easily modified to add other options at the users
# discretion. The optional C program "grabchars", written by Dan Smith
# of Island Graphics, is available via anonymous ftp from Bill Petro for
# users internal to Sun. "grabchars" gets one or more keystrokes from
# the user, without requiring them to hit return. It was written to make
# shell scripts more interactive and simulates "cbreak" features
# available in other programs.

# There are two levels to the cldir script, the sort selection level
# [which can be done alternatively on the command line] and the
# action level [which determines the action to be taken on a specific file].
# Files in a directory may be SORTED by:
# a alphabetical order
# A including "invisible" dot files
# p by pattern using wildcard characters
# s by size, largest first
# L by "larger-than", using the find command
# t by date, oldest first
# r by reverse date, youngest first
# T by age, using the find older-than command

# At the ACTION level, the user determines what to do with the file
# displayed. The user may do nothing and:
# n go on to the next file
# l,. list using the pager of the user's choice
# [configurable in the pager variable]
# p print
# d delete
# M examine by the mail program of choice, if it is a mail folder
# [configurable in the mailer variable]
# f show all filenames for the sort criterion
# : execute a single shell command
# ! create to a subshell in the current working directory
# q quit this iteration of the program
# x bail out and exit the program entirely
# =====
# Comment in the following 3 lines if "grabchars" program is available
grabchars() {
read x; echo $x
}

cldir='basename $0'
mailer="mail" # can be "mail", "Mail", "mush"
pager=more # can be "more", "less", "cat"
shell=csh # can be "sh", "csh", "ksh"
shorthelp="Use (n)ext, (l)ist, (p)rint, (d)elete, (q)uit, e(x)it, or (h)elp"

```



```

    echo "Sorting by $style ..."
    files='/bin/ls -tr'
    ;;
-T|T)    style="Age"
    kind=-T
    echo
    echo -n "Enter how many days ago (ex. 90): "
    read ago
    echo;echo "Sorting by $style ..."
    files='/bin/find . -type f -atime +$ago -print '
    ;;
-u|u)    situation="Unhelpful"
    ;;
-r|r)    style="reverse_date"
    kind=-r
    echo;echo "Sorting by $style ..."
    files='/bin/ls -t'
    ;;
-q|q)    echo;echo Exiting ...;exit ;;
") echo '
Sort Selection:

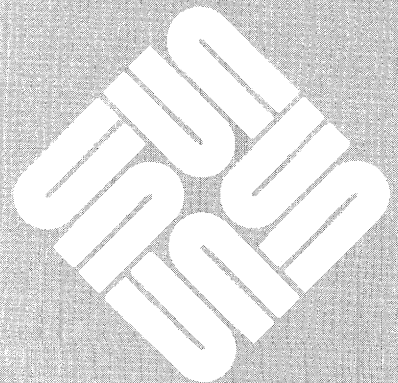
a - Alpha, alphabetical
A - All Alphabetical, including dot files
p - Pattern, by pattern with wildcards
r - Reverse date
s - Size, largest first
L - Larger than, over xx bytes
t - Time, oldest first
T - Age, xx days ago
q - Quit'
    echo -n 'Enter letter of choice: '
    choice='grabchars -b'
    echo
    continue ;;
*) echo;echo "Usage: $0 -[aAprsLtT] "
    exit ;;
esac
if [ $# -ge 1 ]
then
    shift
fi
if [ $# = 0 ]
then
    break
fi
done

echo
# Only echo short help list the first time through, not in subdirectories.
if [ "$situation" != "Unhelpful" ]; then
    echo $shorthelp; echo
fi

```

HARDWARE, CONFIGURATIONS, & UPGRADES

HARDWARE, CONFIGURATIONS, & UPGRADES	1441
Sun386i Ethernet Pins	1441
Sun386i Serial Pins	1442
Software Release Levels	1443



HARDWARE, CONFIGURATIONS, & UPGRADES

Sun386i Ethernet Pins

Sun386i Ethernet Compliance and Pinouts

The *Sun386i Configuration Guide*, February 1989, indicates that Sun386i machines are IEEE 802.3 compliant. However, the Sun386i network interface is electrically Ethernet V2.0 on the built-in Ethernet connection.⁷

The differences between the DIX (Digital-Intel-Xerox) Version 2.0 Ethernet specification and the IEEE 802.3 specification should not affect the operation of the system or the network.

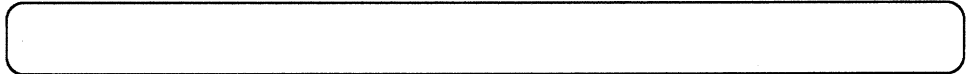
Sun386i Ethernet V2.0 Pinouts

The schematic diagrams show the pinouts of the Ethernet interface on the Sun386i backpanel to be as listed below.

3	Data Out circuit A (Transmit +)
10	Data Out circuit B (Transmit -)
5	Data In circuit A (Receive +)
12	Data In circuit B (Receive -)
2	Control In circuit A (Collision Presence +)
9	Control In circuit B (Collision Presence -)
6	Voltage Common (Power Return)
13	Voltage Plus (Power)
shell	ground/earth
1, 4, 7, 8,	No Connection
11, 14, 15	No Connection

⁷ This article is submitted by Chuck Kollars, Marketing Support Engineering, Boston Development Center.

Sun386i Serial Pins



Sun386i Serial Port Pinouts

The pinouts for the Sun386i serial port are shown in the below list, taken from the page COMM-2 in the Troubleshooting section of the *Field Engineering Handbook - Technical, Volume 1*, dated 5/89.⁸

The connector is RS-232/RS-423, Port A, 25-pin D-Sub, Male.

Pin	Signal
2	TxD
3	RxD
4	RTS
5	CTS
6	DSR
7	GND
8	DCD
15	TxC
17	RxC
20	DTR
22	RI
24	TxCO
25	-5 vdc

⁸ This short subject is submitted by Chuck Kollars, Marketing Support Engineering, Boston Development Center, USA.

Software Release Levels**As of September 22, 1989****Operating Systems**

Product Name	Current Release
SunOS	4.0.3
SunOS SPARCstation 1	4.0.3c
SunOS 386i	4.0.2

Communications Products

Product Name	Current Release
SunLink BSC3270 (SunOS 3.x)	3.0
SunLink BSC3270 (SunOS 4.x)	6.1
SunLink SCP	6.0
SunLink TE100	6.0
SunLink BSCRJE	6.0
SunLink Local 3270	6.1
SunLink SNA3270	6.1
SunLink Peer-to-Peer	6.0
SunLink IR	6.0
SunLink DDN	5.0
SunLink DNI	6.0
SunLink OSI	6.0
SunLink MCP	6.0
SunLink X.25	6.0
SunLink Channel Adapter SCA	6.0
SunLink CG3270	6.0
SunLink MHS	6.0
SunLink HSI	6.0
Notes:	
SunLink release 5.x products are only compatible with SunOS release 3.x.	
SunLink release 6.x products are only compatible with SunOS release 4.x.	

Unbundled Languages

Product Name	Current Release
Sun Modula-2 (Sun-2,3 and SunOS 3.x)	2.0
Sun Modula-2 (Sun-3,4,386i and SunOS 4.x)	2.1
Sun FORTRAN* (Sun-2,3)	1.0
Sun FORTRAN* (Sun-4 and Sys4-3.2)	1.05
Sun FORTRAN* (Sun-2 and SunOS 4.0)	1.1
Sun FORTRAN* (Sun 386i and SunOS 4.0)	1.1R
Sun FORTRAN* (Sun-3,4 and SunOS 4.0)	1.2
SPE for SCLisp 2.1	1.0
Sun Common Lisp-E	1.1
Sun Common Lisp-D	2.1
Sun Common Lisp-D (Sun-3, Sun-4)**	3.0
Cross Compilers (Sun-2,3,4 and SunOS 3.x,Sys4-3.2)	2.0
Cross Compilers (SunOS 4.x, Sun-3,4***)	3.0
Pascal**** (Sun-4 and Sys4-3.2)	1.05
Pascal**** (Sun-2,3,4,386i and SunOS 4.0)	1.1
Notes:	
<p>* The f77 compiler is automatically included with SunOS Release 3.x, which includes SunOS Releases 3.2, 3.4, and 3.5. Sun FORTRAN 1.0 (for Sun-2,3 systems and SunOS 3.x), Sun FORTRAN 1.05 (for Sun-4 systems running Sys4-3.2), Sun FORTRAN 1.1 (for Sun-2,Sun386i systems and SunOS 4.0), and SunFORTRAN 1.2 (for Sun-3,4 and SunOS 4.0) are value-added products that support VMS extensions to the f77 compiler, and must be purchased separately from the SunOS. There is no bundled FORTRAN or Pascal for Sys4-3.2 or SunOS 4.x.</p>	
<p>** Sun Common Lisp-D release 3.0 does not obsolete Sun Common Lisp release 2.1 at this time.</p>	
<p>*** Runs on Sun-3,4 and produces output that also runs on Sun386i.</p>	
<p>**** The pc (Pascal) compiler is automatically included with SunOs Release 3.x, which includes Release 3.2, 3.4, and 3.5. Sun Pascal 1.05 (for Sun-4 systems) and Sun Pascal 1.1 (for Sun-2, Sun-3, Sun-4 and Sun386i systems running SunOS 4.0) are value-added products that support many extensions to the pc compiler, and must be purchased separately from the SunOS.</p>	

Unbundled Graphics

Product Name	Current Release
SunGKS	3.0
SunPHIGS	1.1
Sun58TE	1.0

Unbundled Applications

Product Name	Current Release
SunSimplify	1.1
SunTrac (Sun-2)	1.2
SunTrac (Sun-3,4,386i)	1.3
SunIPC	1.1
Transcript	2.1
SunUNIFY	3.0
PC-NFS	3.0
SunAlis	2.1
SunINGRES (Sun-2 and Sun-3)	5.1

Other Products

Product Name	Current Release
NeWS	1.1
NSE	1.1

TOPS Network Products

Product Name	Current Release
TOPS for the PC	2.1
TOPS for the Sun Workstation (Sun-3, SunOS 3.5)	2.1
TOPS for the Sun Workstation (Sun-3, Sun-4, Sun386i, SunOS 4.X)	2.2
TOPS for the Macintosh	2.1
TOPS NetPrint	2.0

Current Sun Software Products and Release Levels

The preceding tables contain lists of current Sun software products and their respective current release levels.

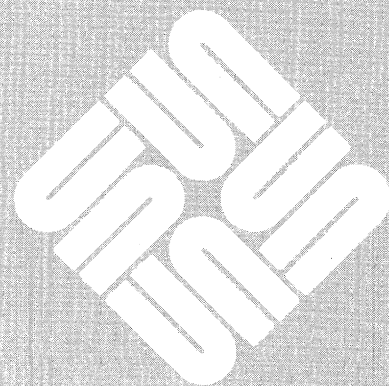
You will note that the Software Technical Bulletin (STB) contains articles from time to time that detail technical changes in a given software product's next available release.

Please contact your sales representative if you decide that you would like to update the release level of a Sun software product you already use, or wish to purchase another product. Use the tables to determine whether your release is the current release level.

These tables appear monthly in the STB for your convenience.

CUMULATIVE INDEX: 1989

CUMULATIVE INDEX: 1989 1451



CUMULATIVE INDEX: 1989

Index

2

2000
SunUNIFY 3.0 dates, 419

5

5210
Micom-Interlan driver upgrade, 416

A

academic software portfolio, 254
access
 netgroups and NFS, 1279
address space
 MS-DOS emulation, 510
addresses
 classes of, 35
 Internet, 35
 network classes, 650
algorithms
 code tuning, 461
alignment
 SPARC porting issues, 266
AnswerLine, 95, 100
Apple
 TOPS, 66
application architectures, 750
 SunOS 4.1, 1046
arch(1)
 and kernel architectures, 752
architectures
 4.1 naming conventions, 1044
 application, 750
 kernel, 750, 751
 kernel and filesystems, 755
 kernel visibility, 753
 small kernels, 764
ARP, 37
AT&T
 OPEN LOOK ordering, 1101
ATbus
 Sun386i drivers, 510
attributes
 SunCGI-SunGKS primitives, 1185
 SunView1 and View2 comparisons, 390
auditing
 SunOS 4.0 security, 774
automounter

automounter, *continued*
 proper YP server binding, 680
awk
 Hackers' Corner introduction, 1203

B

backup
 SNAP hints, 1427
backups
 SNAP and symbolic links, 681
 Sun386i SNAP and restoring files, 1057
 Sun386i SunOS 4.0.1 SNAP, 784
base
 monitor size ordering, 1098
base_level
 SunLink DNI 6.0 installation, 889
Beginner's Guides
 renamed User's Guides, 1062
benchmarking
 corporate, 261
benchmarks
 SDRWAVE and FORTRAN, 685
binaries
 SunOS 4.0.3 prices, 1103
boot
 checkconfig problems, 1056
booting
 diskless client troubleshooting, 1413
 tapeless installs, 949
broadcasting
 subnets, 35
buffer
 memory error message, 787
buffers
 Ethernet management, 86
bug
 online database, 228, 338, 494, 620, 739, 878, 1007, 1131, 1261, 1400
 reporting, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
 reporting in CSD Europe, 229, 339, 495, 621, 740, 879, 1008, 1132, 1262, 1401
 reporting in Intercon, 233, 343, 499, 625, 744, 883, 1012, 1136, 1266, 1405
 reporting in the US, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399

C

- C
 - dbx debugging hints, 703
 - finding zero-divides, 947
 - porting to SPARC, 265
 - c partition
 - whole-disk convention, 912
 - c2
 - SunOS 4.0 security, 768
 - c2conv
 - SunOS 4.0 security, 769
 - cabbling
 - null-modems, 1095
 - serial I/O, 1095
 - cache
 - and device drivers, 55
 - flushing, 51
 - MC68020 on-chip, 42
 - overview, 44
 - performance and moving data, 63
 - Sun-3/200 and Sun-4/200, 42
 - tags, 50
 - variations, 43
 - virtual address, 46
 - caching
 - PC-NFS 3.0 XID, 417
 - calendar
 - 4.0 network traffic, 1411
 - call mapping
 - SunCGI-SunGKS, 1179
 - case study
 - SDRWAVE, 685
 - century
 - SunUNIFY 3.0 dates, 419
 - cg6
 - demonstration programs, 953
 - cgfour
 - moving windows hint, 1343
 - checkconfig
 - problems rebooting, 1056
 - checkmail
 - Hackers' Corner**, 1083
 - checksum
 - Ethernet, 24
 - child processes
 - debugging with dbx, 643
 - classes
 - network addressing, 650
 - client side
 - NFS in depth, 924
 - client-server model
 - Sun386i, 79
 - clients
 - diskless booting troubleshooting, 1413
 - cmdtool
 - disappearing on 3/50s, 1059
 - dying due to signal 1, 1059
 - code
 - tuning hints, 461
 - colormaps
 - bug 1007283, 367
 - colormaps, *continued*
 - flashing, 648
 - managing, 367
 - compatibility
 - Consulting Specials, 354, 1225
 - compilers
 - cross 3.0 announcement, 1285
 - configuration
 - Sun386i, 984
 - Configuration Guide
 - Sun386i, 984
 - configurations
 - 4.1 kernels, 1049
 - SCSI devices, 1037
 - SPARCserver 330 cable lengths, 1043
 - SPARCserver 330 SCSI devices, 1042
 - SPARCstation 1 cable lengths, 1041
 - SPARCstation 1 SCSI devices, 1040
 - Sun-3/80 cable lengths, 1039
 - Sun-3/80 SCSI devices, 1038
 - CONSULT-PROXYARP
 - and Sys4-3.2 subnetting, 521
 - Consulting
 - available Specials, 354, 1225
 - available Sun specials, 354, 1225
 - Sun Germany uucico special, 259
 - controllers
 - SMD-4, 258
 - conventions
 - SunOS 4.1 naming, 1045
 - conversion
 - Sun-to-IBM FP, 469
 - conversions
 - SunView1 to View2 programs, 425
 - coordinates
 - SunGKS 3.0, 1177
 - courses
 - from Sun Educational Services, 349
 - cross compilers
 - 3.0 announcement, 1285
 - cross-referencing
 - Hackers' Corner**, 1203
 - CSD Europe
 - reporting bugs, 229, 339, 495, 621, 740, 879, 1008, 1132, 1262, 1401
 - cursors
 - colormap flashing, 648
 - cylinder groups
 - increasing inodes, 1019
- D**
- daemons
 - 4.0.3 syslogd initialization, 1281
 - printer, 361
 - troubleshooting printer, 569
 - data alignment
 - porting C to SPARC, 266
 - data structures
 - SCSI device drivers, 791
 - database
 - bugs online, 228, 338, 494, 620, 739, 878, 1007, 1131, 1261, 1400

- databases
 - distributed, 1055
 - datagrams
 - fragmentation of, 37
 - reassembly of, 37
 - dates
 - SunUNIFY 3.0 and beyond 2000, 419
 - dbx
 - debugging child processes, 643
 - hints and tips, 703
 - dbxtool
 - hints and tips, 703
 - DC
 - SunGKS 3.0, 1177
 - de-support
 - Sun-2 languages, 415
 - debuggers
 - kernel, 246
 - debugging
 - dbx and child processes, 643
 - demonstration programs
 - cg6, 953
 - demos
 - cg6, 953
 - demultiplexing
 - TCP/IP, 21
 - dependencies
 - software and hardware, 1363
 - dependency tables, 505
 - errata, 1015
 - device coordinates
 - SunGKS 3.0, 1177
 - device drivers
 - 4.0.3 SCSI device driver data structures, 791
 - 4.0.3 SCSI high-level driver theory, 821
 - 4.0.3 SCSI high-low interface, 801
 - 4.0.3 SCSI interface example, 811
 - 4.0.3 SCSI low-high interface, 807
 - 4.0.3 SCSI specification, 791
 - and cache, 55
 - and kernel architectures, 758
 - Sun386i ATbus, 510
 - dialing
 - Sun386i and modems, 1033
 - differential
 - SCSI transmission, 1233
 - direct memory access
 - Sun386i ATbus drivers, 513
 - diskettes
 - used as filesystem tip, 948
 - diskless clients
 - booting troubleshooting, 1413
 - disks
 - 688MByte, 258
 - saving space in **Hackers' Corner**, 1433
 - distributed databases, 1055
 - divide-by-zero
 - finding using dbx, 947
 - DIX
 - and Sun386i Ethernet pinouts, 1441
 - DMA
 - DMA, *continued*
 - Sun386i ATbus drivers, 513
 - DMA channels
 - Sun386i ATbus, 510
 - domain system
 - Internet, 31
 - domains
 - multiple YP, 519
 - domestic kit
 - SunOS 386i 4.0.1, 635
 - DOS
 - enscript hints, 1345
 - expanded memory tip, 1429
 - maximum open files, 257
 - Windows 1.0 announcement, 1156
 - DOS Windows
 - 1.0 announcement, 1156
 - drivers
 - Sun386i ATbus, 510
 - dtree
 - displaying file trees, 260
- ## E
- education
 - available Sun Courses, 349
 - catalog, 778
 - email
 - checkmail **Hackers' Corner**, 1083
 - mush in **Hackers' Corner**, 1349
 - reporting bugs in CSD Europe, 229, 339, 495, 621, 740, 879, 1008, 1132, 1262, 1401
 - reporting bugs in Intercon, 233, 343, 499, 625, 744, 883, 1012, 1136, 1266, 1405
 - reporting bugs in the US, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
 - end of life
 - SunCGI and SunOS 4.1, 564
 - SunCORE and SunOS 4.1, 564
 - end-of-life plan
 - Sun-2 languages, 415
 - enscript
 - hints, 1345
 - environment variables
 - SunView windows, 1170
 - EPS
 - and SunWrite, 631
 - errata, 1139
 - dependency tables, 1015
 - XView, 679
 - error logging, 776
 - error messages
 - Ethernet, 84
 - graphics, 785, 786, 787
 - ioctl, 787
 - printcap tips, 1077
 - errors
 - Ethernet table, 90
 - ioctl #1C, 1053
 - NFS write 13, 559
 - es_file_read error
 - fseek, 1058
 - ESDI shoebox

ESDI shoebox, *continued*
 ordering information, 1099

Ethernet, 24
 buffer management, 86
 error messages, 84
 error table, 90
 header, 24

ethernet
 maximum interfaces, 256

Ethernet
 memory management, 89
 panics, 90
 Sun-3 hardware, 84
 Sun386i pinouts, 1441

Ethernet addresses
 Hackers' Corner, 109

expansion
 DOS expansion tip, 1429

External Data Representation
 NFS in depth, 919

F

FCBs, 257

features
 4.0.3 summary, 897

fhandle
 NFS in depth, 921

file control blocks, 257

file handles
 DOS maximum, 257
 NFS in depth, 921

file locking
 NFS in depth, 931

file translation
 TOPS, 74

file trees
 displaying, 260

files
 maximum DOS open, 257

filesystems
 kernel architectures, 755
 netgroups and NFS access, 1279
 on diskettes tip, 948

find
 displaying file trees, 260

flashing
 colormaps, 648

floating point
 Sun-to-IBM conversion, 469

floppy
 Sun386i format, 911

floppy diskettes
 creating Sun386i UNIX filesystems, 420

flushing
 cache, 51

fonts
 error when missing, 1053
 LaserWriter II, 369

format
 Sun386i floppy disks, 911

FORTTRAN

FORTTRAN, *continued*
 cross-referencing in **Hackers' Corner**, 1203
 dbx debugging hints, 703
 finding zero-divides, 947
 optimizing examples, 1199
 porting hints, 291
 SDRWAVE benchmark, 685
 short warning message, 563
 SunFORTRAN 1.2 announcement, 655
 undefined `_units`, 1058

FP
 Sun-to-IBM conversion, 469

FPU2
 and SunFORTRAN 1.2, 655
 hardware and software support, 852

fragmentation
 datagrams, 37

FreeSpace
 saving disk space in **Hackers' Corner**, 1433

fseek
`es_file_read` error, 1058

FTP, 14

function return values
 porting C to SPARC, 273

G

gateways, 34
 TOPS and PC-NFS, 241

Germany
 uucico Consulting special, 259

graphics
 error messages, 785, 786, 787
`ioctl` error message, 787

groups
 increasing inodes, 1019
 network filesystems, 891
 YP, 891

H

Hacker's Corner
 super kill skill, 299

Hackers' Corner
 checkmail, 1083
 Ethernet addresses, 109
 locate script, 713
 tar -i, 837

handles
 file, 257

hardware
 and software dependencies, 1363
 dependency tables, 505
 Ethernet, 84
 questionnaire, 849
 Sun386i parallel port pins, 851

Hardware Technical Bulletin
 hardware interest, 849

headers
 IP, 23
 octets, 19
 overview, 21

Hints and Tips
 modem installation, 95

- Hints and Tips, *continued*
 - modem problems, 100
 - terminal installation, 95
- hotline@sun.COM*
 - reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
- hotlines
 - world, 225, 335, 491, 617, 736, 875, 1004, 1128, 1258, 1397
- HTB
 - hardware interest, 849
- I**
- IBM
 - FP conversion to Sun, 469
- IBM PC
 - TOPS, 68
- ICMP, 30
- IEEE 302.3
 - and Sun386i Ethernet pinouts, 1441
- ilpr
 - Interleaf to PostScript files, 915
- implementation architectures
 - SunOS 4.1, 1047
- INGRES
 - support transition, 561
- inodes
 - maximum using `mkfs`, 1019
- inputs
 - Sun-CGI-SunGKS, 1191
- installation
 - Sun386i SunOS 4.0.1 remotely, 1021
- installations
 - tapeless, 949
- Intercon
 - hotline, 226, 336, 492, 618, 737, 876, 1005, 1129, 1259, 1398
 - reporting bugs, 233, 343, 499, 625, 744, 883, 1012, 1136, 1266, 1405
- interface
 - 4.0.3 SCSI example, 811
 - 4.0.3 SCSI high-level driver theory, 821
 - 4.0.3 SCSI high-low, 801
 - 4.0.3 SCSI low-high, 807
- interfaces
 - ethernet maximum, 256
- Interleaf
 - to PostScript files, 915
- Internet
 - addresses, 35
 - domain system, 31
 - protocols, 13
- Internet Protocol
 - NFS in depth, 921
 - subnetting, 650
- interrupt channels
 - Sun386i ATbus, 510
- ioctl
 - #1C errors, 1053
 - error messages, 787
- IP, 13
 - headers, 23
 - NFS in depth, 921
 - subnetting, 650
- ISO
 - NFS in depth, 921
- K**
- kernel
 - architectures, 750
 - architectures and kernel-level applications, 753
 - debuggers, 246
- kernel architectures, 751
 - arch(1)*, 752
 - device drivers, 758
 - filesystem layouts, 755
 - kernel-level applications, 753
 - small kernels, 764
 - sun3*, 750
 - sun4*, 750
 - visibility, 753
- kernel configurations
 - SunOS 4.1, 1049
- kernels
 - small pre-configured, 764
 - SunOS 4.0 profiling procedure, 583
- keyboards
 - type 4 dip switches, 1234
- kill(1)*
 - Hacker's Corner, 299
- kit
 - SunOS 386i 4.0.1 domestic, 635
- L**
- languages
 - Sun-2 de-support, 415
- LANs
 - and the Sun386i, 1104
- LaserJet II
 - on Sun386i parallel ports, 653
- LaserWriter II
 - fonts, 369
- LaserWriters
 - troubleshooting, 569
- layering
 - mail, 19
- left shifting
 - textedit bug, 1060
- lex
 - Hackers' Corner** introduction, 1203
- line discipline
 - changing characteristics, 645
- lint
 - use during porting, 265
- Lisp
 - new products, 895
- locate
 - Hackers' Corner script, 713
- locking files
 - NFS in depth, 931
- lockscreen
 - C2 and SunOS 4.0, 526
- log
 - errors, 776
- logging

logging, *continued*
 4.0.3 system daemon, 1281

login
 Sun386i security fix, 783

logintool
 Sun386i security fix, 783

loopback packets
 Sun386i, 893

lpc
 aborting printing daemon, 361
 unreliable daemon killing, 574

lpd
 troubleshooting, 569

lpq
 hints and tips, 1077

lpr
 hints and tips, 1077

ls
 displaying file trees, 260

M

machdep.c
 SunOS 4.0.3 fix, 1286

Macintosh
 TOPS, 66

mail, 15
 layering, 19
 mush in **Hackers' Corner**, 1349
 routing, 33

manual pages
 printing using troff, 418

mapping
 SunCGI-SunGKS calls, 1179

maps
 customized YP, 516

mass storage subsystems
 ordering information, 1099

MC68020
 on-chip cache, 42

memory
 DOS expanded tip, 1429
 textedit window maximum, 1171

memory buffer
 error message, 787

memory management
 Ethernet messages, 89

messages
 valloc failed, 1420

Micom-Interlan 5210
 driver upgrade, 416

mkfs
 maximum inodes per group, 1019
 Sun386i UNIX filesystems, 420

modems
 Hints and Tips, 100
 install Hints and Tips, 95
 null-modem cabling, 1095
 SPARCstation 1, 1061
 Sun386i serial cards, 1033

monitors
 base size ordering, 1098

monitors, *continued*
 corrupted Sun386i vi displays, 1197
 Sun386i, 984

MS-DOS
 address space emulation, 510
 communications software, 1035
 Sun386i and modems, 1035

multicast packets
 Sun386i, 893

mush
Hackers' Corner, 1349

N

name servers, 1153

naming
 4.1 conventions, 1045

NDC
 SunGKS 3.0, 1177

netgroups
 NFS file system access, 1279

netmasks
 default, 650
 subnetting, 650

network
 address classes, 650

Network File System
 client side, 924
 file locking, 931
 filesystem naming, 930
 implementation, 928
 in depth, 919
 overall design goals, 920
 porting experience, 936
 security, 931
 server side, 923
 the protocol, 921
 time skew, 932
 versus RFS, 934

network window systems
 Sun386i, 81

networks
 4.0 calendar traffic, 1411
 filesystem groups, 891
 Sun386i, 77
 Sun386i windows, 81

NeWS 1.1
 errata to RTF, 236

NFS, 16
 client side, 924
 file locking, 931
 filesystem naming, 930
 implementation, 928
 in depth, 919
 netgroups and file system access, 1279
 overall design goals, 920
 porting experience, 936
 security, 931
 server side, 923
 Sun386i, 78
 the protocol, 921
 time skew, 932
 versus RFS, 934

- NFS, *continued*
 - write error 13, 559
- normalized device coordinates
 - SunGKS 3.0, 1177
- O**
- OBD, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
 - change notes, 1271
 - Sun386i bugs added, 253
- octets
 - TCP/IP headers, 19
- ONC
 - Sun386i, 77
- online
 - bugs database, 228, 338, 494, 620, 739, 878, 1007, 1131, 1261, 1400
- Online Bugs Database
 - change notes, 1271
 - Sun386i bugs added, 253
- OPEN LOOK
 - ordering information, 1101
 - STAGE products, 530
- OpenWindows
 - moving windows hint, 1343
- optimizing
 - FORTRAN examples, 1199
- output primitives
 - SunCGI-SunGKS, 1181
- overviews
 - Sun386i on networks, 77
- P**
- packets, 24
 - PC-NFS 3.0 trailers, 255
- padding
 - porting C to SPARC, 270
- panics
 - Ethernet errors, 90
- parallel port
 - Sun386i AT compatibility, 1358
 - Sun386i signals, 851
- parallel ports
 - LaserJet II on Sun386i parallel ports, 653
- parameters
 - passing with SPARC, 275
- partitions
 - whole-disk *c* convention, 912
- passing parameters
 - porting C to SPARC, 275
- PC
 - TOPS, 68
- PC LANs
 - and the Sun386i, 1104
- PC-NFS
 - 3.0 and trailers, 255
 - TOPS gateways, 241
- PC-NFS 3.0
 - XID caching and SunOS 4.0, 417
- performance
 - cache, 63
 - SunOS 4.0 hints, 283
- PIO
 - Sun386i ATbus drivers, 514
- plan
 - Sun-2 language de-support, 415
- plot(flg)*
 - printing files, 1287
- plotters
 - Sun386i serial ports, 829
- PNP
 - Sun386i client install, 421
- portfolio
 - academic software, 254
- porting
 - FORTRAN hints, 291
 - SunCGI to SunGKS 3.0, 1175
 - tutorial, 685
- ports
 - SPARCstation 1 serial and modems, 1061
 - Sun386i parallel and LaserJet IIs, 653
 - Sun386i serial and plotters, 829
 - Sun386i serial voltages, 1357
- PostScript
 - encapsulated (EPS), 631
 - from Interleaf files, 915
- primitive attributes
 - SunCGI-SunGKS, 1185
- primitives
 - SunCGI-SunGKS output, 1181
- printcap
 - hints and tips, 1077
- printer
 - aborting daemon, 361
- printers
 - LaserJet II on Sun386i parallel ports, 653
 - troubleshooting, 569
- printing
 - DOS *enscript* hints, 1345
 - manual pages using *troff*, 418
 - printcap* hints and tips, 1077
 - using *plot(flg)*, 1287
- procedures
 - SunOS kernel profiling, 583
- processes
 - debugging children with *dbx*, 643
- products
 - release levels, 223, 333, 489, 615, 734, 873, 1094, 1224, 1362, 1446
- profiling
 - SunOS 4.0 kernel procedure, 583
- programmed I/O
 - Sun386i ATbus drivers, 514
- programs
 - converting SunView1 to View2, 425
 - porting C to SPARC, 265
- pseudo teletype
 - example program, 587
- pstty*
 - changing characteristics, 645
- ptys*
 - pseudo example program, 587

Q

- questionnaire
 - hardware interest, 849
- queue
 - window error messages, 785

R

- read
 - suntools error, 1172
- real time
 - Sun386i SunOS, 1095
- reassembly
 - datagrams, 37
- reboot
 - checkconfig problems, 1056
- releases
 - software products, 223, 333, 489, 615, 734, 873, 1094, 1224, 1362, 1446
- Remote File System
 - versus NFS, 934
- remote installation
 - Sun386i SunOS 4.0.1, 1021
- Remote Procedure Call
 - NFS in depth, 921
- reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
- resets
 - watchdog, 246
- resistors
 - pull-up, 1357
- restoring files
 - Sun386i SNAP, 1057
- return values
 - porting C to SPARC, 273
- RFS
 - versus NFS, 934
- RGB
 - colormap flashing, 649
- routing
 - mail, 33
- RPC
 - NFS in depth, 921
 - selection service message, 1419
 - source availability, 1029
- RPCSRC 4.0
 - availability, 1029
- RTF
 - NeWS 1.1 errata, 236
- RTI
 - INGRES support, 561
- Rutgers University, 13

S

- scalar
 - algorithm and code tuning, 461
- SCCS
 - installation, 638
 - simplified operations, 638
- sccs
 - the command, 639
- SCSI

SCSI, continued

- 4.0.3 device driver specification, 791
- 4.0.3 high-level driver theory, 821
- 4.0.3 high-low interface, 801
- 4.0.3 interface example, 811
- 4.0.3 low-high interface, 807
- configuration tables, 1037
- SPARCserver 330, 1042
- SPARCserver 330 cable lengths, 1043
- SPARCstation 1, 1040
- SPARCstation 1 cable lengths, 1041
- specification data structures, 791
- Sun hardware implementations, 1233
- Sun-3/80, 1038
- Sun-3/80 cable lengths, 1039
- SDRWAVE
 - case study, 685
- security
 - C2, 768
 - NFS in depth, 931
 - Sun386i SunOS 4.0.1, 783
 - SunOS 4.0, 767
 - SunOS 4.0 c2conv, 769
 - SunOS enhancement, 1167
- selection service
 - RPC calls, 1419
- sendmail
 - expansion fix, 560
- serial cards
 - Sun386i modems, 1033
- serial I/O
 - null-modem cabling, 1095
 - Sun386i SunOS, 1095
- serial port
 - changing characteristics, 645
 - Sun386i pinouts, 1442
 - Sun386i voltages, 1357
- serial ports
 - SPARCstation 1 modems, 1061
 - Sun386i and plotters, 829
- server side
 - NFS in depth, 923
- servers
 - multiply YP, 518
 - name, 1153
- share files
 - SunOS 4.1, 1049
- shifting left
 - textedit bug, 1060
- shoebox
 - ESDI ordering information, 1099
- short
 - FORTTRAN warning message, 563
- silo overflow
 - error messages, 786
- slay
 - killing lpd, 574
- small kernels, 764
- SMD-4 controllers, 258
- SMTP
 - application example, 28
- SNAP

- SNAP, *continued*
 - backup hints, 1427
 - Sun386i YP servers, 680
 - Sun386i client install, 421
 - Sun386i SunOS 4.0.1 backups, 784
 - symbolic links and backups, 681
- SOCK_RDM
 - unimplemented socket, 913
- SOCK_SEQPACKET
 - unimplemented socket, 913
- sockets
 - unimplemented, 913
 - well-known, 25
- software
 - and hardware dependencies, 1363
 - dependency tables, 505
- source
 - SunOS 4.0.3 prices, 1103
- SPARC
 - porting C programs, 265
- SPARCserver 330
 - cable lengths, 1043
 - SCSI configurations, 1042
- SPARCstation 1
 - cable lengths, 1041
 - modems, 1061
 - SCSI configurations, 1040
- Specials
 - compatibility, 354, 1225
- specials
 - Sun Consulting, 354, 1225
- specification
 - 4.0.3 SCSI data structures, 791
 - 4.0.3 SCSI device drivers, 791
 - 4.0.3 SCSI high-level driver theory, 821
 - 4.0.3 SCSI high-low interface, 801
 - 4.0.3 SCSI interface example, 811
 - 4.0.3 SCSI low-high interface, 807
- STAGE
 - OPEN LOOK, 530
 - product announcements, 530
 - SunDraw, 549
 - SunPaint, 542
 - SunWrite, 534
- STB
 - duplication of, 235, 345, 501, 627, 746, 885, 1014, 1138, 1268, 1407
- STREAMS
 - resources, 528
 - SunOS 4.0, 239
- structures
 - SCSI device driver data, 791
- stty(1)
 - changing characteristics, 645
- subnets
 - broadcasting, 35
- subnetting, 650
 - and SunOS Sys4-3.2, 521
 - restrictions, 651
- Sun Academic Software Portfolio
 - Sun Academic Software Portfolio*, 254
- Sun Common Lisp
 - Sun Common Lisp, *continued*
 - new products, 895
 - Sun Consulting
 - available specials, 354, 1225
 - Sun Education
 - catalog, 778
 - Sun Educational Services
 - available courses, 349
 - Sun workstations
 - TOPS, 69
 - sun!hotline*
 - reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
 - sun!stb-editor*, 95, 100, 235, 345, 501, 627, 746, 885, 1014, 1138, 1268, 1407
 - sun!sunbugs*
 - reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
 - Sun-2
 - hardware and software dependencies, 1363
 - language de-support, 415
 - last SunOS supported, 749
 - Sun-3
 - hardware and software dependencies, 1363
 - Sun-3/50s
 - disappearing `cmdtool`, 1059
 - Sun-3/80
 - cable lengths, 1039
 - SCSI configurations, 1038
 - Sun-4
 - hardware and software dependencies, 1363
 - Sun-4/110 TC
 - true color representation, 649
 - sun3
 - 4.1 architecture, 1044
 - kernel architecture, 750
 - Sun386i
 - and PC LANs, 1104
 - ATbus drivers, 510
 - binding to YP servers, 680
 - bugs added to OBD, 253
 - configuration, 984
 - corrupted `vi` displays, 1197
 - floppy format, 911
 - hardware and software dependencies, 1363
 - installing SunUNIFY 3.0, 365
 - loopback/multicast packets, 893
 - network overview, 77
 - NFS, 78
 - ONC, 77
 - parallel port AT compatibility, 1358
 - parallel port pins, 851
 - serial cards and modems, 1033
 - serial port pinouts, 1442
 - serial port voltages, 1357
 - serial ports and plotters, 829
 - SNAP backups and restoring files, 1057
 - SNAP backups and symbolic links, 681
 - SunLink DNI 6.0 installation, 889
 - SunOS 386i 4.0.1 domestic kit, 635
 - SunOS 4.0.1 overview, 523
 - SunOS 4.0.1 performance, 562

- Sun386i, *continued*
 - SunOS 4.0.2 telemarketing ordering, 1273
- Sun386i SunOS
 - 4.0.1 security, 783
 - 4.0.2 backup hints, 1427
- Sun386i SunOS 4.0.1
 - remote installation, 1021
 - SNAP backups, 784
- Sun386i SunOS 4.0.2
 - announcement, 1143
 - telemarketing ordering, 1273
- sun3x
 - 4.1 architecture, 1044
 - kernel architecture, 750
- sun4
 - 4.1 architecture, 1044
 - kernel architecture, 750
- sun4c
 - 4.1 architecture, 1044
 - kernel architecture, 750
- sunbugs@sun.COM*
 - reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130, 1260, 1399
- SunCGI
 - call mapping to SunGKS, 1179
 - end-of-life and SunOS 4.1, 564
 - inputs, 1191
 - output primitives, 1181
 - porting to SunGKS 3.0, 1175
 - primitive attributes, 1185
- SunCore
 - C functions and SunPHIGS, 1323
- SunCORE
 - end-of-life and SunOS 4.1, 564
- SunCore
 - to SunPHIGS translation guide, 1291
- SunDraw 1.0
 - announcement, 549
 - STAGE product, 530
- SunFORTRAN
 - 1.2 announcement, 655
- SunGKS
 - 3.0 porting from SunCGI, 1175
 - 3.0 workstations, 1176
 - call mapping to SunCGI, 1179
 - inputs, 1191
 - new concepts, 1176
 - output primitives, 1181
 - primitive attributes, 1185
- SunGKS 2.2.1
 - announcement, 566
- SunINGRES
 - support transition, 561
- SunLink
 - 4.0.3 upgrade bug, 1054
- SunLink DNI 6.0
 - installation, 889
- SunOS
 - 3.5 and type 4 keyboards, 1234
 - 386i 4.0.1 domestic kit, 635
 - 386i performance, 562
 - 386i 4.0.1 remote installation, 1021
- SunOS, *continued*
 - 4.0 c2conv, 769
 - 4.0 calendar traffic, 1411
 - 4.0 security, 767
 - 4.0 security auditing, 774
 - 4.0.3 announcement, 749
 - 4.0.3 feature summary, 897
 - 4.0.3 machdep.c fix, 1286
 - 4.0.3 syslogd initialization, 1281
 - 4.0.3 upgrade bug, 1054, 1168
 - 4.0.3 upgrade paths, 853
 - 4.1 and SunCGI end-of-life, 564
 - 4.1 and SunCORE end-of-life, 564
 - 4.1 Beginner's Guides renamed, 1062
 - 4.1 directory layout, 1044
 - 4.1 kernel configurations, 1049
 - 4.1 naming conventions, 1045
 - C2 4.0 security, 768
 - determining version, 363
 - security enhancement, 1167
 - SPARCstation1 OS prices, 1103
 - Sun-2 language de-support, 415
 - Sun386i 4.0.1 overview, 523
 - Sun386i 4.0.1 security, 783
 - Sun386i 4.0.2 announcement, 1143
 - Sun386i 4.0.2 backup hints, 1427
 - Sun386i 4.0.2 telemarketing ordering, 1273
 - Sun386i null-modem cabling, 1095
 - Sun386i serial I/O, 1095
 - Sys4-3.2 and subnetting, 521
 - Sys4-3.2 tapeless installs, 949
- SunOS 3.5
 - type 4 keyboard settings, 1234
- SunOS 3.x.x
 - finding Ethernet addresses, 109
- SunOS 4.0
 - C2 lockscreen, 526
 - calendar traffic, 1411
 - error logging differences, 776
 - finding Ethernet addresses, 109
 - kernel profiling procedure, 583
 - PC-NFS 3.0 XID caching, 417
 - performance hints, 283
 - starting suntools, 1169
 - STREAMS, 239
- SunOS 4.0.3
 - announcement, 749
 - feature summary, 897
 - machdep.c fix, 1286
 - SCSI device driver data structures, 791
 - SCSI driver specification, 791
 - SCSI high-level driver theory, 821
 - SCSI high-low interface, 801
 - SCSI interface example, 811
 - SCSI low-high interface, 807
 - SPARCstation1 OS prices, 1103
 - syslogd initialization, 1281
 - upgrade bug, 1054, 1168
- SunOS 4.1
 - Beginner's Guides renamed, 1062
 - directory layout, 1044
 - kernel configurations, 1049
 - naming conventions, 1045

- SunOS Sys4-3.2
 - tapeless installs, 949
 - SunPaint 1.0
 - announcement, 542
 - STAGE product, 530
 - SunPHIGS
 - C functions and SunCore, 1323
 - from SunCore translation guide, 1291
 - upgrading reasons, 1292
 - SunSimplify
 - installing with SunUNIFY 3.0, 365
 - sunttools
 - missing font errors, 1053
 - read errors, 1172
 - starting under 4.0, 1169
 - SunUNIFY 3.0
 - bug 1016117, 365
 - dates beyond 2000, 419
 - documentation updates, 365
 - installing on Sun386i, 365
 - sunupgrade(8)*
 - bug, 1168
 - bug with SunLink, 1054
 - SunVideo
 - announcement, 1065
 - applications, 1070
 - features, 1066
 - hardware, 1067
 - software, 1069
 - SunView
 - missing font errors, 1053
 - window error, 1170
 - SunView1
 - attribute comparison with View2, 390
 - converting programs to View2, 425
 - SunWrite
 - and EPS, 631
 - SunWrite 1.0
 - announcement, 534
 - STAGE product, 530
 - survey
 - hardware interest, 849
 - symbolic links
 - SNAP backups, 681
 - Sys4-3.2
 - and subnetting, 521
 - syslogd
 - SunOS 4.0.3 initialization, 1281
- T**
- TAAC
 - software release 2.3, 1031
 - true color representation, 649
 - TAAC 2.3
 - announcement, 1031
 - right to use, 1032
 - tables
 - hardware/software dependencies, 505
 - software release levels, 223, 333, 489, 615, 734, 873, 1094, 1224, 1362, 1446
 - tags
 - cache, 50
 - tapeless installations, 949
 - tar
 - damaged tapes, 837
 - Hackers' Corner, 837
 - scavenging files, 837
 - tcopy
 - tputil Consulting Differences, 244
 - TCP, 13
 - TCP/IP
 - demultiplexing, 21
 - references, 38
 - telemarketing
 - ordering Sun386i SunOS 4.0.2, 1273
 - teletype
 - pseudo example program, 587
 - TELNET, 14
 - terminals
 - install Hints and Tips, 95
 - textedit
 - left shifting bug, 1060
 - maximum window memory, 1171
 - theory of operation
 - 4.0.3 SCSI high-level drivers, 821
 - time skew
 - NFS in depth, 932
 - TOPS
 - file translation, 74
 - for Apple Macintosh, 66
 - for IBM PC, 68
 - for Sun workstations, 69
 - installation requirements, 75
 - Macintosh operation, 71
 - PC operation, 72
 - PC-NFS gateways, 241
 - product overview, 66
 - technical support, 243
 - use of, 71
 - tputil
 - Consulting special, 244
 - copying SunOS release tapes, 244
 - trailers
 - with PC-NFS 3.0, 255
 - TranScript
 - troubleshooting, 569
 - translation
 - Core-PHIGS overview, 1292
 - SunCore to SunPHIGS guide, 1291
 - TOPS files, 74
 - transmission
 - SCSI single-ended, 1233
 - trees
 - displaying files, 260
 - troff
 - printing manual pages, 418
 - troubleshooting
 - diskless client booting, 1413
 - Ethernet errors, 90
 - LaserWriters, 569
 - TranScript, 569
 - true color representation
 - colormap flashing, 649
 - tuning

tuning, *continued*
 coding hints, 461
 tutorial
 porting, 685
 type 4 keyboards
 switch settings and SunOS 3.5, 1234

U

UDP, 30
 NFS in depth, 921
 Ultrix 2.x
 with PC-NFS 3.0 trailers, 255
 undefined `_units`
 workaround, 1058
 upgrades
 4.0.3 and bug, 1168
 4.0.3 and SunLink bug, 1054
 paths to SunOS 4.0.3, 853
 User Datagram Protocol
 NFS in depth, 921
 User's Guides
 SunOS 4.1, 1062
 uucico
 Sun Consulting Germany special, 259
 uucp
 Sun Germany uucico special, 259

V

valloc
 failed error message, 1420
 variables
 SunView environment, 1170
 VAX
 and PC-NFS 3.0 trailers, 255
 vector
 algorithm and code tuning, 461
 version
 determining SunOS level, 363
 VFS
 NFS in depth, 919, 925
 vi
 corrupted Sun386i display, 1197
 video
 SunVideo announcement, 1065
 SunVideo applications, 1070
 SunVideo features, 1066
 SunVideo hardware, 1067
 SunVideo software, 1069
 View2
 attribute comparison with SunView1, 390
 converting programs from SunView1, 425
 XView errata, 679
 virtual address
 cache, 46
 Virtual File System
 NFS in depth, 919, 925
 VMEbus
 custom boards, 914
 Revision B and Sun-3s, 914

W

watchdog resets, 246
 WC
 SunGKS 3.0, 1177
 well-known sockets, 25
 windows
 colormap flashing, 648
 Windows
 DOS 1.0 announcement, 1156
 windows
 error messages, 785
 faster moving and `cgfour`, 1343
 maximum `textedit` memory, 1171
 SunView error, 1170
 workstation
 SunGKS 3.0, 1176
 workstations
 TOPS, 69
 world coordinates
 SunGKS 3.0, 1177
 world hotlines, 225, 335, 491, 617, 736, 875, 1004, 1128, 1258,
 1397
 write errors
 NFS 13, 559
 write-back cache, 43
 write-through cache, 43

X

X.25
 usage hints, 289
 XDR
 NFS in depth, 919
 porting C to SPARC, 272
 XID
 PC-NFS 3.0 and SunOS 4.0, 417
 XNS
 NFS in depth, 921
 XView
 errata, 679

Y

yellow pages
 customized maps, 516
 hints, 516
 multiple servers, 518
 name servers, 1153
 serving multiple domains, 519
 YP
 4.0 calendar slave traffic, 1411
 customized maps, 516
 filesystem groups, 891
 hints, 516
 multiple servers, 518
 name servers, 1153
 serving multiple domains, 519
 YP servers
 Sun386i binding, 680
 YP slaves
 4.0 calendar traffic, 1411
`ypset(8)`
 fix available, 565

Z

zero-divides

finding using dbx, 947

Bulk Rate
U.S. Postage
PAID
Racine, WI
Permit No. 957

Systems for Open Computing™

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 37-29639

**For U.S. Sales Office
locations, call:**
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor, Green Lane
Bagshot, Surrey GU19 5NL
England
0276 51440
TLX 859017

Australia: (02) 413 2666
Canada: 416 477-6745
France: (1) 40 94 80 00

Germany: (089) 95094-0
Hong Kong: 852 5-8651688
Italy: (39) 6056337
Japan: (03) 221-7021
Korea: 2-7802255
Nordic Countries: +46 (0)8 7647810
PRC: 1-8315568
Singapore: 224 3388
Spain: (1) 2532003
Switzerland: (1) 8289555
The Netherlands: 3133501234

Taiwan: 2-7213257
UK: 0276 62111

**Europe, Middle East, and Africa,
call European Headquarters:**
0276 51440

**Elsewhere in the world,
call Corporate Headquarters:**
415 960-1300
Intercontinental Sales

